UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

Diseño y generación semi-automática de patrones adaptables para el reconocimiento de entidades

Autor:
Mónica Marrero Llinares

Directores:
Sonia Sánchez Cuadrado
Jorge Morato Lara

DEPARTAMENTO DE INFORMÁTICA

Leganés, marzo de 2013

TESIS DOCTORAL

DISEÑO Y GENERACIÓN SEMI-AUTOMÁTICA DE PATRONES ADAPTABLES PARA EL RECONOCIMIENTO DE ENTIDADES

Autor: Mónica Marrero Llinares

Directores: Sonia Sánchez Cuadrado
Jorge Morato Lara

Firma del Tribunal Calificador:

Firma

Presidente: Juan Llorens Morillo

Vocal: Roberto Carniel

Secretario: Rafael Valencia García

Calificación:

Leganés, de de

Índice de contenidos

ÍNDICE DE FIGURAS III
ÍNDICE DE TABLAS VII
ÍNDICE DE CÓDIGOS XI
GLOSARIO XIII
ABSTRACT 1
RESUMEN 3
CAPÍTULO 1 INTRODUCCIÓN 5
CAPÍTULO 2 RECONOCIMIENTO DE ENTIDADES NOMBRADAS 9
2.1. Introducción ... 9
2.2. Técnicas de Reconocimiento de Entidades Nombradas ... 21
2.3. Atributos y recursos ... 33
2.4. Representación de patrones ... 43
2.5. Tipos de herramientas ... 48
2.6. Principales foros de evaluación ... 52
CAPÍTULO 3 EVALUACIÓN DE HERRAMIENTAS NER 59
3.1. Introducción ... 59
3.2. Herramientas NER evaluadas ... 61
3.3. Metodología ... 65
3.4. Eficacia de las herramientas ... 67
3.5. Errores más frecuentes ... 70
3.6. Discusión ... 74
3.7. Conclusiones ... 75
CAPÍTULO 4 FALACIAS Y RETOS EN RECONOCIMIENTO DE ENTIDADES NOMBRADAS 77
4.1. Introducción ... 77
4.2. Evolución del significado de NE ... 78
4.3. ¿Qué es una Entidad Nombrada? ... 80
4.4. Implicaciones en la evaluación del Reconocimiento de Entidades ... 84
4.5. Implicaciones en las herramientas NER ... 87
4.6. ¿Está NER realmente resuelta? ... 88
4.7. Foros actuales ... 92
4.8. Retos y oportunidades en NER ... 94
4.9. Conclusiones ... 96
CAPÍTULO 5 MODELADO DE PATRONES PARA EL RECONOCIMIENTO DE ENTIDADES: GRAMÁTICAS IEG 99
5.1. Introducción ... 99
5.2. Características de los modelos de representación ... 101
5.3. Gramáticas adaptadas a la Extracción de Información: IEG ... 104
5.4. Complejidad computacional de una gramática IEG ... 108
5.5. Adaptación a *Speech Recognition Grammar Specification* ... 117
5.6. Conclusiones ... 119
CAPÍTULO 6 APRENDIZAJE INDUCTIVO DE GRAMÁTICAS IEG 121

6.1. Introducción 121
6.2. Trabajos relacionados 123
6.3. Generación de gramáticas IEG: descripción general del método 127
6.4. Fase I: reducción de caracteres 131
6.5. Fase II: creación de funciones 133
6.6. Inducción de reglas 135
6.7. Conclusiones 145

CAPÍTULO 7 EVALUACIÓN Y RESULTADOS **147**
7.1. Introducción 147
7.2. Configuración 150
7.3. Tarea A. Generación de gramáticas IEG 155
7.4. Tarea A. Generación de gramáticas IEG: efectividad y coste con mínimo número de semillas 172
7.5. Tarea A. Generación de expresiones regulares 175
7.6. Tarea B. Anotación de corpus 178
7.7. Conclusiones 184

CAPÍTULO 8 CONCLUSIONES **187**
CAPÍTULO 9 TRABAJOS FUTUROS **191**
CONCLUSIONS **193**
FUTURE WORK **195**
APÉNDICE. EJEMPLO DE FUNCIONAMIENTO DEL MÉTODO DE GENERACIÓN DE PATRONES **197**
Selección de semillas 197
Fase I: Atributos de granularidad carácter 198
Fase II. Atributos de granularidad token 201
Gramática IEG resultante 203

BIBLIOGRAFÍA **205**

Índice de figuras

Figura 1. Jerarquía de entidades de nombre propuesta por Sekine. 11
Figura 2. Ejemplo de árbol de decisión para detección de entidades. 23
Figura 3. Procesos de meta y mutual bootstrapping (Riloff, Jones 1999). 30
Figura 4. Ejemplo de *infobox* correspondiente a un artículo sobre la ciudad de Innsbruck. A la izquierda aparece el código fuente. 42
Figura 5. Relaciones entre DBpedia y otras fuentes de datos (Fuente: wiki.dbpedia.org/Interlinking). 43
Figura 6. Precision, recall y F en identificación completa y clasificación de entidades. 67
Figura 7. Relación entre la F obtenida en identificación completa y en clasificación. 68
Figura 8. Relación de la F en la identificación y clasificación considerando identificación completa o parcial. 68
Figura 9. Ejemplo de autómata en cascada para el reconocimiento de nombres de persona [Chiticariu et al., 2010b]. Cada fase representa un autómata finito que reconoce como alfabeto de entrada los símbolos especificados en "Input". Las reglas contienen acciones (→) que indican los símbolos de salida, que serán los de entrada del autómata de la siguiente fase. 102
Figura 10. Ejemplo de salida de CPSL y JAPE [Chiticariu et al., 2010b]. Basado en la gramática de la Figura 9. 104
Figura 11. Construcción de un AFN-ε con un único estado de aceptación y sin arcos que entren al estado inicial o que salgan del estado de aceptación. Autómatas para la unión, concatenación y clausura. R y S son lenguajes regulares. El resultado, $LR \cup LS$, $L(R) \cap L(S)$, $L(R*)$ es también un lenguaje regular [Hopcroft et al., 2006]. 109
Figura 12. Autómatas equivalentes para la unión, concatenación y clausura con transiciones épsilon añadidas. 110
Figura 13. Autómata T con funciones asociadas. La transición ε de entrada provocará el almacenamiento de la posición en la cadena de entrada. El autómata sólo transitará al estado de aceptación (o a un nuevo autómata concatenado o que lo contenga) si se cumplen las funciones F_R sobre la cadena de entrada hasta la posición actual. 110
Figura 14. Tabla construida por el algoritmo CYK [Hopcroft et al., 2006]. 115
Figura 15. Ejemplo de gramática (izq.) y tabla correspondiente construída por el algoritmo CYK (dcha.) para la cadena de entrada w = baaba. 115
Figura 16. Método de generación de gramáticas IEG. 128
Figura 17. Rectas de regresión para la F-measure dependiendo del contexto para el corpus seminarios y jobs. 151
Figura 18. Rectas de regresión para el número de tokens validados dependiendo del contexto para el corpus seminarios y jobs. 152
Figura 19. Distribución de la F-measure para cada valor de umbral en el corpus de seminarios y jobs. Los rombos muestran los intervalos de confianza al 95% alrededor de la media. 153

Figura 20. Rectas de regresión, a modo meramente ilustrativo, para la F-measure dependiendo del umbral por cada tipo de entidad del corpus seminarios y jobs. 154
Figura 21. Rectas de regresión para el número de tokens validados dependiendo del umbral para el corpus seminarios y jobs. 155
Figura 22. Relación de F-measure del conjunto de entidades y ejemplos validados obtenidos sobre el corpus de seminarios para cada una de las diferentes ejecuciones. La línea aproxima la F por número de ejemplos validados a partir de las medias obtenidas por número de semilla. 157
Figura 23. Relación de F-measure del conjunto de entidades y tokens validados obtenidos sobre el corpus de seminarios para cada una de las diferentes ejecuciones. La línea aproxima la F por número de tokens validados a partir de las medias obtenidas por número de semilla. 158
Figura 24. Relación entre la precision (P) y el recall (R) en función del número de ejemplos validados para cada número de semillas iniciales. 159
Figura 25. Relación entre la efectividad en términos de F-measure y coste de anotación en el corpus de seminarios para cada una de las fases. Los colores indican las semillas iniciales. La línea continua indica la aproximación de la F-measure en fase I, mientras que la discontinua corresponde a la fase II. 159
Figura 26. Comparación de la F-measure obtenida en la fase I sobre el corpus seminarios con el algoritmo presentado en [Li et al., 2008a] y utilizando como unidad de anotación los tokens (línea roja continua), fragmentos de 11 tokens (línea roja discontinua) y documentos (línea roja punteada). 160
Figura 27. Comparación de la F-measure obtenida en la fase II sobre el corpus seminarios con el algoritmo presentado en [Li et al., 2008a] y utilizando como unidad de anotación los tokens (línea roja continua), fragmentos de 11 tokens (línea roja discontinua) y documentos (línea roja punteada). 161
Figura 28. F-measure obtenida en la fase II en relación a los ejemplos validados por tipo de entidad. 161
Figura 29. Precision y recall obtenidas en la fase II para los tipos *etime* y *stime*. 162
Figura 30. Relación entre los resultados de Li et al. para las diferentes unidades de anotación y los obtenidos al añadir como atributo la entidad *stime* para la identificación de *etime* y a la inversa. 163
Figura 31. Relación entre la F-measure obtenida en la fase II sobre el corpus seminarios (sustituyendo los tipos *etime* y *stime* por *time*) con el algoritmo presentado en [Li et al., 2008a] y utilizando como unidad de anotación los tokens (línea roja continua), fragmentos de 11 tokens (línea roja discontinua) y documentos (línea roja punteada). 165
Figura 32. Relación de F-measure y ejemplos validados sobre el corpus jobs. Los colores indican el número de semillas iniciales, y la línea es una aproximación sobre la media de las ejecuciones por número de semillas. 167
Figura 33. Relación de F-measure y tokens validados sobre el corpus jobs. Los colores indican el número de semillas iniciales, y la línea es una aproximación sobre la media de las ejecuciones por número de semillas. 167
Figura 34. Relación entre la precision (P) y el recall (R) en función del número de ejemplos validados para cada número de semillas iniciales, mostradas en código de colores. 168

Figura 35. Relación entre la efectividad en términos de F-measure y coste de anotación en el corpus jobs para cada una de las fases. Los colores indican las semillas iniciales. En línea continua se representa la aproximación de la F-measure de la fase I, en discontinua la de la fase II. 168
Figura 36. Comparación de la F-measure obtenida en la fase II sobre el corpus jobs con el algoritmo presentado en [Li et al., 2008a] y utilizando como unidad de anotación los tokens (línea roja continua), fragmentos de 11 tokens (línea roja discontinua) y documentos (línea roja punteada). 170
Figura 37. Diagramas de caja y medias (rombos) obtenidas de la F-measure por entidad en el corpus jobs. 171
Figura 38. Relación de efectividad y coste al aplicar un único proceso con un número variable de semillas iniciales (Sección 7.3.1) o dos procesos consecutivos partiendo de 2 semillas iniciales. 174
Figura 39. Jobs: relación de efectividad y coste al aplicar un único proceso con un número variable de semillas iniciales (Sección 7.3.2) o dos procesos consecutivos partiendo de 2 semillas iniciales. 175
Figura 40. Tasas de precision y recall para los tipos de entidades del corpus seminarios. 181
Figura 41. Selección de entidades iniciales en el corpus no anotado. Se revisan los documentos y al localizar una entidad se presiona "Add Selected Text". Estas entidades (que aparecerán marcadas en naranja) serán las semillas para iniciar el proceso de aprendizaje de patrones. 197
Figura 42. Reglas candidatas resultantes (en naranja y amarillo). Aparece separado el contexto de la entidad. 198
Figura 43. Proceso de validación de potenciales entidades localizadas por el sistema en el corpus no anotado. Las entidades aparecen señaladas en verde. El usuario pude pulsar sobre ellas para indicar que no son válidas (aparecen entonces marcadas en rojo). También es posible modificar los límites de cada entidad en el cuadro de edición inferior. 199
Figura 44. Nuevas reglas candidatas resultantes (en amarillo) de los patrones posicionales mejor alineados tanto en el contexto como en la propia entidad. 199
Figura 45. Reglas candidatas resultantes (en naranja, amarillo, azul y rojo). En gris se muestran las reglas creadas en fases previas. 200
Figura 46. Reglas candidatas resultantes (enazul, naranja y amarillo). En gris se muestran las reglas creadas en fases previas. 201
Figura 47. Reglas candidatas resultantes (en azul, amarillo, naranja y rojo). En gris se muestran las reglas creadas en fases previas. 201
Figura 48. Reglas candidatas y finalmente creadas (en azul, naranja y amarillo). 202
Figura 49. Reglas candidatas y finalmente creadas (en amarillo y naranja). 202

Índice de tablas

Tabla 1. Descripción de la tarea de ranking de entidades (Cheng, Yan, & Chang, 2007)..... 15
Tabla 2. Patrones iniciales de contenido como semillas para aprendizaje en NER (Collins, Singer 1999)..... 30
Tabla 3. Ejemplo de predicados unarios y binarios, y patrones del sistema *KnowItAll*. 32
Tabla 4. Regla generada a partir de la combinación del patrón NP1 "such as" NPList2, y el predicado City. En realidad en el sistema estas reglas son representadas mediante una gramática en notación BNF. 33
Tabla 5. Atributos de identificación de entidades en el sistema NERUA [Ferrández et al., 2005]..... 35
Tabla 6. Regla obtenida por (LP)2. Los elementos en negrita forman parte de la regla obtenida a partir de la generalización de la secuencia de texto. 45
Tabla 7. Corpus utilizado en CoNLL 2003. El corpus en inglés procede de noticias periodísticas de *Reuters*, y el corpus en alemán procede de noticias periodísticas del periódico *Frankfurter Rundshau*. 53
Tabla 8. Resultados sobre el corpus de test de las herramientas que compitieron en CoNLL 2003. 54
Tabla 9. Resultados de los sistemas en la competición ACE'08 para la tarea de Reconocimiento Local de Entidades (*Local Entity Detection and Recognition*) sobre corpus en inglés (los porcentajes negativos son debidos al sistema de penalización utilizado). 55
Tabla 10. Ejemplo de pregunta de entrada en INEX Entity-Ranking Track. 56
Tabla 11. Características de las herramientas. El símbolo de interrogación indica que no ha sido posible encontrar información al respecto. 64
Tabla 12. Características analizadas y número de entidades en el documento con cada característica. 65
Tabla 13. Tipos de entidades distribuidas en el documento de evaluación. 66
Tabla 14. Resultados por tipo de entidad. 69
Tabla 15. Influencia de los atributos en el porcentaje de errores en identificaciones parciales, identificaciones completas y clasificaciones. Los números indican el porcentaje de errores cuando el atributo no se cumple y cuando sí. Diferencias significativas al * 0.05, ** 0.01 y *** 0.001. 72
Tabla 16. Influencia de los atributos en el porcentaje de identificaciones incorrectas (parcial o completa) por herramienta. Los números indican el porcentaje de identificaciones incorrectas cuando el atributo no se cumple y cuando sí. Diferencias significativas al * 0.05, ** 0.01 y *** 0.001. 73
Tabla 17. Ejemplo de NEs extraídas de la jerarquía de Sekine (SH), y de los corpus anotados o las guías de MUC, CoNLL03, ACE y GENIA (GEN). Los símbolos de interrogación indican dependencia del contexto. 80
Tabla 18. Ejemplos obtenidos de los corpus o guías de anotación (con referencias a la norma exacta seguida) de MUC, CoNLL03 y ACE. Se muestra el diferente modo de anotación de una misma entidad y las diferencias de criterio

respecto a la identificación como NE (I), sus límites (B) y la categoría asignada (C). 84
Tabla 19. Relaciones aproximadas entre los tipos de entidad reconocidos en MUC7, CoNLL03 y ACE08. 85
Tabla 20. Ejemplo de etiquetado de la frase *new york's la guardia airport […]* de acuerdo a la ACE. NOM: nominal, NAM: name, FAC: facility, GPE: geo-political entity, PopCenter: population centre, SPC: specific referent. 86
Tabla 21. Mapeo de los principales elementos de ABNF (RFC5234) a SGRS. 118
Tabla 22. Categorías Generales de Unicode 130
Tabla 23. Agrupación propia de categorías Unicode 130
Tabla 24. Patrones posicionales potenciales en una gramática con las entidades $Rp1 = 972 - 830165$ y $Rp2 = 972 - 563410$ 138
Tabla 25. Patrones posicionales potenciales en una gramática con las entidades $Rp1 = 972 - 243 - 0120$ y $Rp2 = 972484 - 9330$ 139
Tabla 26. Aplicación de la heurística al caso de un terminal "a" que aparece en diferentes posiciones en cada Rpi. Para cada posición alternativa de "a" en los diferentes Rpi (posiciones 0, 5, 6 y 7) se crea un *path* con las posiciones de "a" más cercanas. Estos *path* son: (0,0,0,0,0,0), (5,5,6,7,7,6) y (7,7,6,7,7,6). 140
Tabla 27. Agrupación de las distribuciones de F-measure para diferentes valores de contexto según el test de Tukey-Kramer. Las distribuciones en el grupo A son las que obtienen valores más altos para F-measure, mientras que en las de B ocurre lo contrario. Los grupos A y B son significativamente diferentes. El grupo AB no es significativamente diferente de A y de B. 152
Tabla 28. Valores medios de efectividad (F-measure, precisión y recall) y coste (fragmentos de texto validados y su equivalente en tokens) resultantes en el corpus de seminarios para diferente número de semillas iniciales. 157
Tabla 29. Valores medios de F-measure y suma de los costes de anotación para todos los tipos de entidad. 160
Tabla 30. F-measure media y suma de ejemplos y tokens validados para todos los tipos de entidad por número de semillas, considerando atributos adicionales para *stime* y *etime*. 163
Tabla 31. Relación de efectividad obtenida (F-measure, precision y recall) y coste (ejemplos validados y el equivalente en tokens) por número de semillas para la entidad *time*. 164
Tabla 32. F-measure media y suma de ejemplos y tokens validados para los tipos *location*, *speaker* y *time* por número de semillas. 165
Tabla 33. Valores medios de efectividad (F-measure, precision y recall) y coste (fragmentos de texto validados y tokens equivalentes) por tipo de entidad en el corpus jobs para diferente número de semillas iniciales. 166
Tabla 34. Valores medios de efectividad (F-measure, precision y recall) y coste (fragmentos de texto validados y tokens equivalentes) por tipo de entidad en el corpus jobs (eliminando *phone*) para diferente número de semillas iniciales 169
Tabla 35. Valores medios de F-measure y suma de los costes para el reconocimiento de todos los tipos de entidad. 169
Tabla 36. Resultados al aplicar un único proceso con un número variable de semillas iniciales o dos procesos consecutivos partiendo de 2 semillas iniciales. 173

Tabla 37. Resultados al aplicar un único proceso con un número variable de semillas iniciales o dos procesos consecutivos partiendo de 2 semillas iniciales con las entidades *time, speaker* y *location.* 174
Tabla 38. Jobs: resultados al aplicar un único proceso con un número variable de semillas iniciales o dos procesos consecutivos partiendo de 2 semillas iniciales..... 175
Tabla 39. Semillas utilizadas para los tipos de entidad *time, phone* y *post_date.* No se encuentran en el corpus, y las que están en negrita además presentan patrones diferentes a las que se encuentran en él. 177
Tabla 40. Medias e intervalos de confianza con tres semillas propias. 177
Tabla 41. Medias e intervalos de confianza con tres semillas externas. 178
Tabla 42. Relación Recall y coste de anotación medios para el corpus Seminarios. 179
Tabla 43. Relación Recall y coste de anotación medios, considerando que el coste de obtener una semilla equivale a la anotación de un documento completo. 180
Tabla 44. Relación precision , recall y costes de anotación (generación y validación) para cada tipo de entidad en el corpus seminarios tomando 5 semillas. 181
Tabla 45. Relación Recall y coste de anotación medios para el corpus Jobs. 182
Tabla 46. Relación Recall y coste de anotación medios, considerando que el coste de obtener una semilla equivale a la anotación de un documento completo. 182
Tabla 47. Tasas de precision y recall para cada tipo de entidad en el corpus jobs tomando 5 semillas. Se muestra el coste de anotación de generación de los patrones sumado al coste de validar las identificaciones, expresado todo ello en número de documentos aproximados a validar. 183

Índice de códigos

Código 1. Información semi-estructurada extraída del Wall Street Journal. 12
Código 2. Plantilla para clasificar la información. 13
Código 3. *Concept Node* identificado por AutoSlog para la localización de objetivos de bombas. 44
Código 4. Ejemplo de patrón en Whisk para identificar eventos de reemplazo de puestos de trabajo: @Passive indica la forma de la palabra, *Person* es una sub-expresión, @succeed es un *stem* mientras *F se refiere a cualquier cadena pero de la misma categoría sintáctica. 44
Código 5. Ejemplo de gramática CPSL. 46
Código 6. Fragmento de código de una ontología de extracción para monitores en Ex. 47
Código 7. Ejemplo de topic para la tarea *Slot Filling* en TAC. 57
Código 8. Ejemplo de topic en el TREC Entity Track. 58
Código 9. Gramática IEG para la representación de nombres de persona tanto en modo directo como inverso. 107
Código 10. Gramática IEG para el reconocimiento de números que comienzan por 91 y están en negrita. La nomenclatura de expresiones regulares es la adoptada en el framework Microsoft DotNet. 108
Código 11. Gramática G de tipo IEG con producciones épsilon y su transformación a G' sin producciones épsilon: $LG' = LG - \{\varepsilon\}$ 113
Código 12. Transformación de una gramática G' sin producciones épsilon, a una gramática G'' en la Forma Normal de Chomsky: $LG'' = LG'$ 114
Código 13. Algoritmo CYK para el reconocimiento de un lenguaje independiente de contexto. 116
Código 14. Algoritmo CYK para el reconocimiento de un lenguaje independiente de contexto con una gramática IEG. Los cambios introducidos respecto al algoritmo CYK aparecen en un recuadro. 116
Código 15. Ejemplo de gramática descrita en SRGS [Hunt & McGlashan, 2004]. 117
Código 16. Extracto de DTD del estándar SRGS. En negrita las modificaciones introducidas para incorporar funciones. 119
Código 17. Ejemplo de patrón generado por la herramienta Whisk. Person y Position son subexpresiones previamente almacenadas. El prefijo "@" indica el valor resultante de un stemmer. "*F" indica repetición de cero o más tokens pero con la misma categoría morfo-sintáctica. 124
Código 18. Gramática G' inicial generada a partir de las semillas. Su símbolo inicial es S, mientras que su alfabeto es $\Sigma c1$. 132
Código 19. Estado de la gramática G'tras aplicar el algoritmo de inducción con el sub-alfabeto $\Sigma c1$. Se han creado las reglas $Rc1$ y $Rc2$. En el proceso se ha incorporado un nuevo ejemplo positivo propuesto por el algoritmo y validado por el usuario, que pasa a formar parte de la gramática. Los terminales en los cuerpos de producción son transformados a $\Sigma c2$, excepto los contenidos en las reglas inducidas. 133
Código 20. Gramática G' una vez finalizada la fase de análisis de atributos a nivel carácter, donde nuevas reglas de inducción, $Rc3$ y $Rc4$ han sido creadas. El

proceso de tokenización crea los no terminales $Rt1$-$Rt4$, que agrupan en tokens los cuerpos de producción de $Rp1$-$Rp4$. 134
Código 21. Gramática G'' procedente de G' al aplicar el atributo longitud sobre los tokens en los cuerpos de producción (excepto en los de las reglas inducidas). El algoritmo de inducción sobre G'' puede generar una regla que comprimiría el valor "9" en $Rt1$ - $Rt4$. Esa nueva regla sería aplicada en forma de función en G' a los no terminales de $Rt1$ a $Rt4$. 134
Código 22. Algoritmo de generación de gramáticas IEG 135
Código 23. Heurística para la generación de *paths*. 141
Código 24. Algoritmo de generación de gramáticas IEG con el proceso de clustering incorporado. Los cambios introducidos respecto al algoritmo del Código 22 aparecen en un recuadro. 143
Código 25. Algoritmo de inducción de reglas (RuleInduction) con la incorporación de aprendizaje activo. 144
Código 26. Expresión regular para la captura de números de teléfono (entidad *phone*) en el corpus jobs. 148
Código 27. Expresión regular para la captura de indicaciones de hora en el corpus seminarios. La complejidad de la misma se debe a la necesidad de evitar la confusión con otras entidades numéricas. 164
Código 28. Gramática IEG resultante del método de generación de patrones. Por simplicidad, las reglas generadas en el cuerpo de los no terminales A y B han sido omitidas y en los cuerpos de producción se utilizan directamente los símbolos utilizados en los ejemplos y otras simplificaciones entre corchetes. 203

Glosario

- AL: *Active Learning* / Aprendizaje Activo
- Atributo: propiedad de un objeto de estudio que sirve para describirlo. En relación a las entidades nombradas como objeto de estudio, ejemplos de atributos son los tokens que lo forman, su información sintáctica, la pertenencia a listados, etc. Según el tamaño de la unidad a la que se refiera el atributo, se habla de diferentes niveles de granularidad. Por ejemplo, de mayor a menor nivel de granularidad tendríamos los atributos de carácter, token, palabra, frase, documento, etc.
- CPSL: *Common Pattern Specification Language*
- Exhaustividad / recall: se diferencia el concepto (exhaustividad) de la medida (*recall)*.
- Flexibilidad: el concepto de potencia asociado a los patrones como modelos se entenderá como la capacidad para representar diferentes atributos capaces de describir los elementos a reconocer.
- F-measure (también conocida como F-score): medida de efectividad basada en precision y recall. Se abreviará en ocasiones como *F*.
- IE: *Information Extraction* / Extracción de Información.
- Minería de opiniones: *Opinion Mining*
- NE: *Named Entity* / Entidad Nombrada. Se abreviará en ocasiones como *entidad*. Existen varias definiciones de este concepto, que se discuten en el Capítulo 4. A partir de ellas se deriva la definición de Entidad Nombrada como cualquier semántica que pueda considerarse de interés en un dominio de aplicación, que es la adoptada en este trabajo.
- NER: *Named Entity Recognition* / Reconocimiento de Entidades Nombradas. Se abreviará en ocasiones como *Reconocimiento de Entidades*.
- NLP: *Natural Language Processing* / Procesamiento del Lenguaje Natural.
- Patrón: Modelo que sirve de muestra para sacar otra cosa igual (ejemplo de uso: generación de patrones) / conjunto de elementos que se repiten de forma predecible (ejemplo de uso: localización de patrones).

- Población de ontologías: *Ontology Population*
- Potencia: el concepto de potencia asociado a los patrones como modelos se entenderá como la capacidad para representar diferentes estructuras sintácticas.
- Precisión / precision: se diferencia el concepto (precisión) de la medida (*precision)*.
- Sistemas de búsqueda de respuesta: *question answering systems*

Abstract

The task of Named Entity Recognition (NER) facilitates information management and is useful in other areas like Semantic Annotation, Question Answering, Ontology Population and Opinion Mining.

According to the results from some evaluation forums though, NER may be considered a solved task. This dissertation digs into these evaluations and shows that they seemed stuck to the recognition of typical entities for which annotated resources are usually available. This contrasts with the current diversity of entity types and domains of application. The main contribution of this work is the design of a method to recognize entities that is more consistent with the lack of annotated corpora for any required type of entity and in any domain. The designed method integrates the following aspects:

- Transparency: readable patterns with a high level of standardization.
- Flexibility: possibility to incorporate different types of features capable of describing entities or their context.
- Power: recognition of different language structures within documents.
- Cost: use of a small set of entities as initial seeds and active learning techniques to guide the user through the annotation process.
- Effectiveness: competitive effectiveness rates compared to the state of the art in terms of precision and recall.

The method is evaluated with two public annotated corpora with different types of entities, and compared with related works found in the scientific literature.

Resumen

La tarea de Reconocimiento de Entidades Nombradas (NER) facilita la gestión de información y tiene utilidad en otras áreas, como Anotación Semántica, Sistemas de Búsqueda de Respuesta, Población de Ontologías y Minería de Opiniones.

Pero de acuerdo a los resultados de algunos foros, el área de NER podría considerarse resuelta. La tesis profundiza en la evaluación del área y muestra que parece haberse estancado en el reconocimiento de entidades típicas, para las que habitualmente existen recursos anotados. Esto contrasta con la diversidad de tipos de entidad y dominios de aplicación actuales. Este trabajo contribuye con el diseño de un método para el reconocimiento de entidades más consecuente con el problema de no disponer de corpus anotados para cualquier tipo de entidad requerida y sobre cualquier dominio. El método diseñado integra los siguientes aspectos:

- Transparencia: patrones legibles y con alto grado de estandarización.
- Flexibilidad: posibilidad de incorporar diferentes tipos de atributos capaces de describir las entidades o su contexto.
- Potencia: reconocimiento de diferentes estructuras del lenguaje en los documentos.
- Coste: uso de un pequeño conjunto de entidades como semillas iniciales y técnicas de aprendizaje activo para guiar al usuario en el proceso de anotación.
- Efectividad: tasas de efectividad competitivas en relación al estado del arte, medidas en términos de precisión y exhaustividad.

Los resultados obtenidos son evaluados mediante el uso de corpus públicos anotados con diferentes tipos de entidades, y comparados con otros trabajos relacionados en la literatura científica.

Capítulo 1
Introducción

Between 2009 and 2020, the information in the Digital Universe will grow by a factor of 44. Yet the staffing and investment to manage the Digital Universe will grow by a factor of 1.4. This should make it clear to CIOs [Chief Information Officer] and business executives that much of the next 10 years of their careers will be spent dealing with the challenge of the mismatch of these growth rates. [One of the issues will be] developing tools for search and discovery of information as the Digital Universe expands, including finding ways to add structure to unstructured data though metadata, automatic content tagging, and pattern recognition. -The Digital Universe Decade, Are You Ready?[Gantz & Reinsel, 2010]

La información digital es uno de los principales activos de cualquier organización, y su buena gestión puede determinar el éxito y el liderazgo en muchos sectores. La gestión de la información supone no sólo el tratamiento de la información generada en la organización, sino también la posibilidad de hacer uso de la información que nos rodea, y que puede suponer ventajas competitivas: estudio de mercados, análisis de opiniones, seguridad, vigilancia tecnológica, etc. Es en este entorno donde cobran mayor sentido las labores de extracción de información.

La Extracción de Información, a diferencia de la recuperación de información, nos permite identificar y acceder directamente a aquellos elementos de información de nuestro interés, reduciendo la cantidad de información que necesitamos leer. Una de las áreas de la

Extracción de Información es la llamada *Named Entity Recognition* (NER), que tiene por objetivo la identificación de semánticas relevantes en un texto, y es necesaria por tanto para la posterior construcción de relaciones o escenarios. Algunas de las aplicaciones inmediatas de NER son el procesamiento del lenguaje natural, la automatización en tareas de anotación semántica y la población de ontologías, aunque se utiliza también en sistemas de búsqueda de respuesta y minería de opiniones, por citar algunos.

El Reconocimiento de Entidades Nombradas ha sido evaluado en grandes foros a nivel internacional. Las elevadas tasas de efectividad observadas en dos de estos foros, *Message Understanding Conference (MUC)* y *Conference on Natural Learning (CoNLL),* entre finales de los 90 y principios de los 2000 podrían justificar que la tarea esté resuelta, y así ha llegado a afirmarse en la literatura. La verificación de esta afirmación lleva a la realización de un primer trabajo que consiste en el análisis de las características, efectividad y errores de herramientas NER de propósito general, contenido en el Capítulo 3. Los resultados obtenidos, con tasas de efectividad muy inferiores a las obtenidas en MUC y CoNLL, así como la diversidad en cuanto al número y tipo de las entidades reconocidas llevan a profundizar no sólo en la cuestión de la efectividad lograda en este ámbito, sino también en sus objetivos, condicionados por la definición del mismo término *Named Entity*. El Capítulo 4 analiza estas cuestiones y muestra que las evaluaciones de NER en el ámbito internacional han sido realizadas usando un limitado conjunto de entidades prácticamente invariable año tras año, y con corpus pequeños en relación a los utilizados en otras áreas de la recuperación de información. Estos factores, entre otros, han limitado la evolución del área y llevado a conclusiones erróneas al generalizar los resultados. Se concluye que la afirmación de que se trata de un área resuelta no está justificada y que las unidades de evaluación utilizadas en la evaluación del área deben ser extendidas y variadas. Estas necesidades se traducen también en una mejora de la capacidad de adaptación de los sistemas NER a nuevos tipos de entidades y dominios, de modo que sean capaces de responder a las necesidades actuales de los usuarios y ayudar en la creación de recursos para la evaluación.

Entre los requisitos para lograr métodos más adaptables se encuentran la potencia expresiva y flexibilidad de los patrones para el reconocimiento de entidades, que permitan la adaptación a las diversas estructuras y atributos que pueden caracterizar a las entidades y su contexto. El esfuerzo para su generación es otro factor a considerar. Los métodos que requieren corpus anotados previos implican no sólo un esfuerzo en anotación, sino además conocimiento por parte del usuario del proceso de anotación y habilidad para localizar y equilibrar anotaciones positivas y negativas adecuadas.

Ante este escenario se plantea como objetivo el diseño de un método de generación de patrones para el reconocimiento de entidades capaz de adaptarse a los nuevos tipos de entidades y dominios requeridos por usuarios y aplicaciones, donde habitualmente no existen corpus anotados o no son de libre acceso. Para ello en el Capítulo 5 se diseña un modelo de representación de patrones potente y flexible que, en la línea de transparencia exigida por los nuevos sistemas de extracción de información, es además comprensible y basado en estándares para facilitar su adopción. El Capítulo 6 describe un método semi-automático de generación de patrones basados en este modelo, que a partir de un pequeño número de entidades iniciales guía al usuario en el proceso de generación.

Finalmente, el Capítulo 7 presenta el diseño y resultados de los experimentos realizados. Se utilizan diversos tipos de entidades y corpus para probar una relación de efectividad y coste de anotación competitiva frente a métodos similares. Las conclusiones muestran el grado de cumplimiento de los objetivos propuestos, y en trabajos futuros se proponen mejoras frente a los límites del trabajo realizado e investigaciones futuras que puedan suponer contribuciones adicionales.

Capítulo 2
Reconocimiento de Entidades Nombradas

Our knowledge can be only finite, while our ignorance must necessarily be infinite –Karl Popper

2.1. Introducción

2.1.1. Origen

El término "Entidad Nombrada" (*Named Entity*- NE) fue acuñado en la sexta conferencia MUC (*Message Understanding Conference*) [MUC-6 Program Committee, 1995] [Grishman & Sundheim, 1996]. Estas conferencias fueron iniciadas en 1987 bajo el patrocinio de la agencia norteamericana de defensa DARPA, con el objetivo de fomentar los métodos de Extracción de Información (IE) a partir de textos no estructurados. El objetivo era clasificar semánticamente la información de interés, que sobre todo en las primeras conferencias, tenía tintes militares. Ejemplos típicos de esta serie de conferencias era determinar en un texto el agente, tiempo, causa y lugar de un evento. Ya en 1995, con la sexta conferencia MUC, se ve la importancia de la identificación de nombres de personas, organismos y localizaciones, y de expresiones numéricas como el tiempo y las cantidades. Esta tarea es

denominada Reconocimiento de Entidades Nombradas (en adelante Reconocimiento de Entidades o NER), constituyéndose como un importante campo dentro de la IE.

En investigaciones previas ya se habían tratado problemas similares. Éste es el caso del trabajo presentado en 1991 por Lisa F. Rau et al. [1991], donde se describe un sistema para extraer y reconocer nombres de compañías mediante el uso de heurísticas y reglas elaboradas *ad hoc* a partir de observaciones. Sin embargo, es desde finales de los 90 cuando se acelera el ritmo de las publicaciones en este ámbito y comienzan a surgir además foros de evaluación relacionados con esta y otras tareas de extracción de información: HUB [Chinchor et al., 1998], IREX: Information Retrieval and Extraction Exercise [Sekine & Isahara, 2000], CoNLL: Conference on Natural Language [Sang, 2002][Sang & Meulder, 2003], ACE: Automatic Content Extraction [Doddington et al., 2004], etc.

2.1.2. Definición

No existe una definición comúnmente aceptada del término "Entidad Nombrada". Esta cuestión se ha puesto de manifiesto en la literatura científica: el concepto de NE apareció en un entorno de aplicaciones de Procesamiento del Lenguaje Natural (NLP) y está lejos de estar asentado y ser lingüísticamente claro [Borrega et al., 2007]. Algunos trabajos formulan el problema del reconocimiento de entidades como reconocimiento de nombres propios en general [Petasis et al., 2000]. De hecho, los tipos de entidades más investigadas hasta el momento son fundamentalmente los nombres propios de personas y organismos, y los referentes a localizaciones, que constituyen un conjunto de categorías acuñado con el término *Enamex* desde MUC-6. Sin embargo, esta definición no parece ser suficiente porque algunas características comunes de los nombres propios, como la falta de inflexiones o determinantes, la ausencia de significado léxico y el uso de mayúsculas son insuficientes para describir las entidades de nombre [Borrega et al., 2007]. Este es el caso de las fechas (*Timex*), cantidades (*Numex*), nombres de productos, enfermedades, o días de la semana, por citar algunos.

Otra definición del concepto de NE se apoya en la clasificación de objetos desconocidos en jerarquías que sean útiles para la resolución de problemas particulares [Alfonseca & Manandhar, 2002]. Esta aproximación ha sido seguida por ejemplo en la construcción de la clasificación de tipos de entidades útiles en sistemas de *Question Answering* propuesta por Sekine (Figura 1).

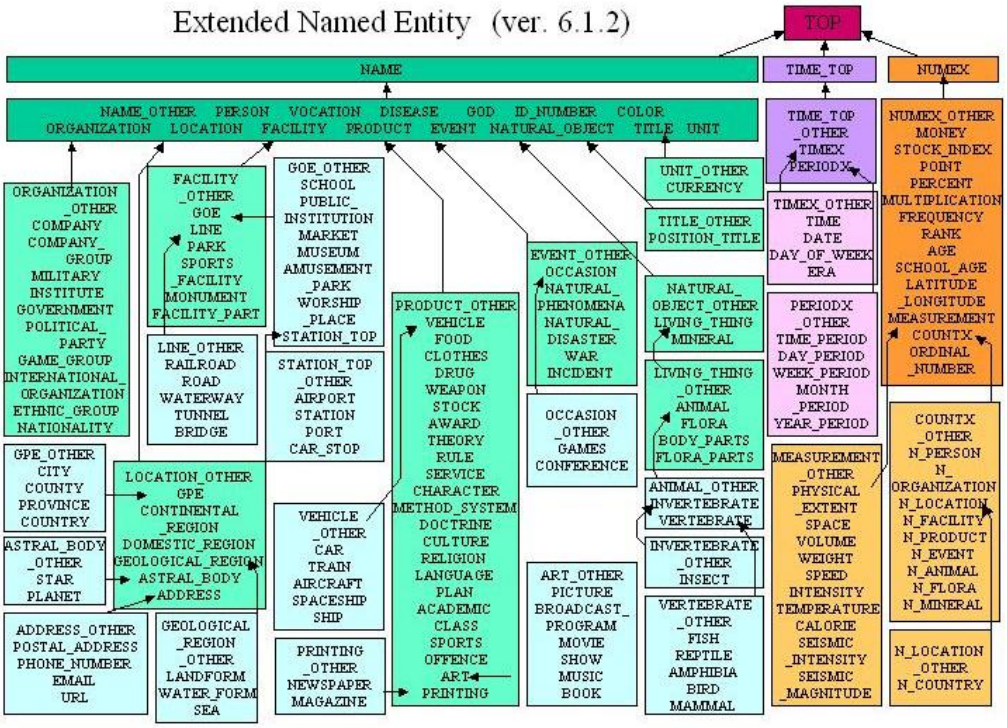


Figura 1. Jerarquía de entidades de nombre propuesta por Sekine[1].

Por último, el concepto de "designador rígido" acuñado por Kripke y procedente de la Filosofía del Lenguaje, también ha sido utilizado para definir las entidades de nombre. En concreto, en [Nadeau & Sekine, 2007] se especifica que el término *named* pretende restringir las entidades a aquellas para las que uno o varios designadores rígidos permanecen como referentes. Kripke define un designador rígido como aquel que designa el mismo objeto en cualquier mundo posible en el que ese objeto exista[2]. Por ejemplo, "Richard Nixon" sería un designador rígido, en oposición a "Presidente de los Estados Unidos", que no lo es porque la entidad a la que se refiere cambia cada pocos años [Kripke, 1980].

[1] http://nlp.cs.nyu.edu/ene/

[2] *A designator d of an object x is rigid if it designates x with respect to all possible worlds where x exists, and never designates an object other than x with respect to any possible world.*

2.1.3. Relación con otros campos

Extracción de Información

La Extracción de Información (IE) "es habitualmente definida como el proceso de estructurar y combinar selectivamente los datos que están explícita o implícitamente en uno o más documentos en lenguaje natural"[3] [Moens, 2006]. También podemos encontrarla definida como el proceso de derivar datos no ambiguos, a partir de textos en lenguaje natural, en servicio de algún tipo de necesidad de información precisa previamente especificada [Cunningham, 2005].En cualquier caso, se trata de un proceso que implica la clasificación semántica de unidades de información.

Un ejemplo de extracción de información lo podemos encontrar en [GATE Project Team, 2010]: a partir del siguiente extracto del *Wall Street Journal* (Código 1), la tarea consiste en rellenar una plantilla con la información correspondiente a cada evento, tal y como se muestra en el ejemplo (Código 2).

```
DOC>
<DOCID> wsj93_050.0203 </DOCID>
<DOCNO> 930219-0013. </DOCNO>
<HL>    Marketing Brief: @  Noted.... </HL>
<DD> 02/19/93 </DD>
<SO> WALL STREET JOURNAL (J), PAGE B5 </SO>
<CO>    NYTA </CO>
<IN> MEDIA (MED), PUBLISHING (PUB) </IN>
<TXT> <p>
New York Times Co. named Russell T. Lewis, 45, president and general
manager of its flagship New York Times newspaper, responsible for all
business-side activities. He was executive vice president and deputy
general manager. He succeeds Lance R. Primis, who in September was
named president and chief operating officer of the parent.
</p> </TXT>
</DOC>
```

Código 1. Información semi-estructurada extraída del Wall Street Journal.

En este ejemplo vemos cómo la extracción de entidades se convierte en un paso previo para categorizar la información que se nos pide según la plantilla. Pero la información recuperada va más allá de la extracción de entidades y establece relaciones entre ellas, configurando de este modo la semántica asociada a un evento. Es por ello que la tarea de

[3] *Information Extraction is usually defined as the process of selectively structuring and combining data that are explicitly stated or implied in one or more natural language documents.*

NER se engloba dentro de la IE, pero no implica necesariamente que todos los procesos de IE supongan extracción de entidades.

Son diversas las tareas que pueden incluirse en el área de IE. Entre ellas cabe destacar el reconocimiento de roles semánticos, la identificación de relaciones entre entidades, la resolución de co-referencia, la identificación de atributos de entidades, etc. Estas tareas ya se evaluaban en las conferencias MUC, y se continuaron considerando en las competiciones ACE hasta 2008, con tareas de reconocimiento de entidades, cantidades, expresiones temporales, relaciones entre entidades y eventos. En realidad las cuatro primeras tareas contribuyen al reconocimiento de eventos, y se realizan utilizando tipos predefinidos de entidades: instalación, entidad geopolítica, localización, organización, persona, vehículo y arma.

```
<SUCCESSION-1>
    ORGANIZATION : "New York Times"
    POST         : "president"
    WHO_IS_IN    : "Russell T. Lewis"
    WHO_IS_OUT   : "Lance R. Primis”
</SUCCESSION-1>
```

Código 2. Plantilla para clasificar la información.

Recuperación de Información

La Recuperación de Información es la localización de material (usualmente documentos) de naturaleza no estructurada (generalmente texto en lenguaje natural) procedente de colecciones digitalizadas que satisface una necesidad de información [Manning et al., 2008]. Difiere de la extracción de información en que supone la recuperación de una colección de documentos relevantes para unas necesidades de información dadas. Por el contrario, la IE persigue la extracción de datos dentro de los documentos.

Debido al creciente volumen de información digital en las colecciones, la recuperación de documentos relevantes puede desbordar al usuario. La consecuencia es una mayor necesidad de herramientas capaces de reducir la cantidad de texto que es necesario leer por parte del usuario para localizar la información relevante. La IE es una de las tecnologías clave para facilitar esta tarea, fundamentalmente el reconocimiento de entidades y relaciones entre ellas [Moens, 2006]. En este sentido, Crestan y Loupy [2004] mostraron que

la extracción de entidades ayuda a los usuarios a realizar búsquedas en grandes colecciones de modo más rápido y eficiente.

Minería de Datos

La minería de datos es el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos [Witten & Frank, 2005]. Por tanto, dos son los retos principales [Orallo et al., 2005]:

- Trabajar con grandes volúmenes de datos.
- Usar técnicas adecuadas para analizarlos y extraer conocimiento novedoso y útil.

También es posible hablar de la inferencia de información no explícita a partir de textos, que constituye un área conocida como *Text Mining* [Moens, 2006]. En cualquier caso, las tareas de extracción de información y reconocimiento de entidades difieren de la minería de datos fundamentalmente en que ambas tienen por objetivo extraer conocimiento explícito, y no inferir nuevo conocimiento.

2.1.4. Aplicaciones

Anotación Semántica

La anotación semántica va más allá de la anotación acerca del contenido de los documentos para formalmente identificar en ellos conceptos y relaciones. Por ejemplo, es posible relacionar el término "París" con una ontología que lo identifique como instancia de "ciudad" y lo relacione así con la instancia "Francia" de la clase "país", evitando de esta forma la ambigüedad que pudiera surgir sobre ese término. Estas anotaciones pueden mejorar la recuperación de información y además favorecen la interoperabilidad [Uren et al., 2006]. Pero es necesario automatizar la anotación para conseguir así procesos escalables ante grandes colecciones [Reeve & Han, 2005]. Esta anotación es típicamente implementada con técnicas de Extracción de Información, entre las que se encuentra el Reconocimiento de Entidades Nombradas.

Sistemas de Búsqueda de Respuestas

Los sistemas de búsqueda de respuestas (*Question Answering –QA- Systems*) proporcionan como resultado una respuesta concreta a una consulta. Las técnicas NER son frecuentemente usadas en este tipo de sistemas como medio para facilitar la selección de respuestas. Un estudio de Srihari and Li [2000] muestra que efectivamente los sistemas IE, y más concretamente los sistemas NER, son necesarios para proporcionar respuestas a la mayoría de los tipos de preguntas. De hecho, en la competición TREC-8 (*Text REtrieval Conference*) para evaluar este tipo de sistemas, un 80% de las preguntas requerían como respuesta una entidad (preguntas del tipo quién, cuándo y dónde, que se responden respectivamente con entidades de persona, fecha y localización).

Algunos autores incluso consideran que las entidades de nombre responden habitualmente a las cinco preguntas típicas de un entorno periodístico (*Five Ws)*: *what*, *where*, *when*, *who*, *why*, capaces de resumir un evento o noticia. Esta característica ha sido aprovechada para capturar este tipo de entidades, a partir de la observación de la correlación existente entre ser una entidad y aparecer en un momento determinado en diferentes fuentes de noticias periodísticas [Shinyama & Sekine, 2004].

Tabla 1. Descripción de la tarea de ranking de entidades (Cheng, Yan, & Chang, 2007)

Colección	Colección de tipos de entidades $\varepsilon = \{E_1, \dots, E_N\}$ sobre Colección de documentos $\mathcal{D} = \{d_1, \dots, d_n\}$
Entrada	Pregunta $q(\langle E_1, \dots E_m \rangle) = \alpha(E_1, \dots E_m, k_1, \dots, k_l)$ donde α es un llamado *tuple pattern*, $E_i \in \varepsilon$ y k_j es el patrón (keyword) para la localización de cada tipo de entidad.
Salida	Lista ordenada $t = \langle e_1, \dots, e_m \rangle$ donde $e_i \in E_i$, ordenada por $\text{Score}(q(t))$, la puntuación de t para esa pregunta q

La ordenación por relevancia en las respuestas, cuando éstas son entidades, se denomina Ranking de Entidades (*Entity Rank)* (Tabla 1). Existen competiciones relacionadas con esta tarea, siendo una de las más conocidas *INEX Entity-Ranking track*, que comenzó en el 2007 [Demartini et al., 2009]. Se trata de un área que comprende conocimiento de QA, de recuperación de información y de reconocimiento de entidades.

Asignación de Temporalidad

La asignación de temporalidad consiste en identificar entidades temporales y relacionarlas con otras entidades, o más frecuentemente, con eventos. Por ejemplo, las fechas de las conferencias o las fechas de nacimiento de las personas. Smith [2002] usa la detección de eventos para localizar en mapas las guerras que han tenido lugar y las fechas en las que se produjeron. Mani y Wilson [2000] detectan expresiones temporales de noticias con el objetivo de ordenarlas cronológicamente. Para ello se utiliza un conjunto básico de patrones y su refinamiento mediante aprendizaje automático.

Población de Ontologías (*Ontology Population*)

La Web Semántica actualmente depende de las ontologías que dan estructura a la información en la Web. Es crucial por tanto la proliferación de las mismas [Maedche & Staab, 2001]. Una de las tareas vitales es el enriquecimiento de ontologías existentes mediante la incorporación automatizada de elementos, habitualmente instancias. El Reconocimiento de Entidades Nombradas ayuda en la tarea de identificar instancias de los conceptos requeridos. Un ejemplo de este tipo de sistema es la herramienta KnowItAll [Etzioni et al., 2005].

Análisis de Opiniones (Opinion Mining o Sentiment Analysis)

Los últimos años han marcado la expansión de la Web Social, donde la gente expresa libremente sus opiniones en una amplia variedad de materias. A la hora de tomar una decisión, cada vez más gente busca opiniones expresadas en la Web acerca de aquello que le interesa y muchas veces basa su decisión final en la información encontrada (Pang B., Lee L. 2009). El análisis de opiniones ayuda en esta labor, clasificando la información en función del tipo de opinión mostrada mediante el uso de procesamiento del lenguaje natural y extracción de información, entre otras técnicas. Un ejemplo sería la búsqueda de opiniones negativas acerca de un producto concreto, para asegurarnos de que no presenta características no deseables y asegurarnos de que no dará problemas. Por ejemplo, Popescu y Etzioni [2005] desarrollan un sistema llamado OPINE para la extracción de atributos de

productos y el análisis de opiniones respecto a ellos. Este sistema se asienta sobre el sistema KnowItAll para la población de ontologías [Etzioni et al., 2005].

Procesamiento del Lenguaje Natural

Una variedad de tareas de procesamiento de lenguaje natural, como la anotación sintáctica y la resolución de coreferencia, se benefician de la existencia de información acerca de entidades de nombre [Brill & Resnik, 1994][Fisher et al., 1995]. La información semántica sobre ciertas palabras ayuda por ejemplo a evitar su división si se trata de palabras compuestas, y puede facilitar la identificación de agentes en la frase, evitando errores en el procesamiento.

2.1.5. Problemas relacionados

Ambigüedad

Etzioni se refiere a su sistema KnowItAll, citado anteriormente, como un extractor de entidades de nombre (*NE Extraction System*), matizando la diferencia entre extracción y reconocimiento de entidades [Etzioni et al., 2005]. Con esto quiere indicar que el sistema no está pensado para resolver la ambigüedad que el reconocimiento de una entidad puede presentar en un documento, sino tan sólo para generar listados de entidades. Sin la resolución de la ambigüedad, un sistema no puede llevar a cabo de forma precisa la tarea de reconocimiento de entidades en documentos, aunque sí la de generación automática de listados para población de ontologías con los fines anteriormente comentados (e.g. anotar erróneamente "John F. Kennedy" como persona en un documento donde se cita para referirse al aeropuerto del mismo nombre puede tener consecuencias negativas en la interpretación de la semántica del documento, pero situar el término "John F. Kennedy" bajo la categoría "persona" en una ontología no deja de ser correcto).

La ambigüedad puede ocurrir tanto entre entidades, como entre entidades y sustantivos comunes. En el primero de los casos, una misma entidad puede pertenecer a más de una categoría. Por ejemplo, el término "Ford" podría ser el nombre de una persona, el de una

organización o el de una marca de coches. Para este problema, en Petasis et al. [2001] se propone que al menos una ocurrencia de la entidad debe aparecer en un contexto donde la categoría correcta es evidente. Por ejemplo, en el contexto "Mr. Ford", está claro que "Ford" se refiere a una entidad de persona. Este tipo de palabras que anteceden o preceden a una entidad y proporcionan información acerca de ella son llamadas disparadores (*triggerwords).* Se usan habitualmente en las herramientas NER, incorporándolas bien manualmente o bien como atributos para la creación de modelos de aprendizaje automático (ver Sección 2.3.1). Entre las más conocidas se encuentran los prefijos personales (e.g. Sr, Sra.), los profesionales (e.g. Dr, Profesor) y los términos habituales en nombres de compañías (e.g. Corp, SA).

Un problema de ambigüedad diferente, aunque también del tipo entidad-entidad, se da cuando una entidad tiene referentes diferentes, aunque del mismo tipo. El caso más estudiado es el que trata la discriminación entre personas con el mismo nombre. Este problema está muy relacionado con la resolución de alias, que veremos posteriormente. De aquí surge una tarea denominada *People Search Task (WePS)* en SemEval-2007 [Artiles et al., 2007], que consiste en agrupar páginas web según la identidad de las personas a las que se refieran. Para eso se proporciona un corpus anotado manualmente con 4722 páginas web y 79 nombres.

Por otro lado la ambigüedad puede presentarse entre términos que pueden funcionar como entidades o no dependiendo del contexto [Da Silva et al., 2004]. Un ejemplo es el término "Rosa", que en español puede tener distinto significado, bien como entidad de persona o bien como sustantivo referente a un tipo de flor o a un color. Para evitar este problema Mikheev [1999] propone una heurística en donde se asume que un término con la primera letra en mayúsculas es una entidad, a menos que aparezca al principio de una frase, en un documento completamente en minúsculas, o en una sección donde todas las palabras comiencen por mayúsculas (e.g. en un título o epígrafe). Otra aproximación para resolver el mismo problema la encontramos en los trabajos de Da Silva [2004], donde se compara la frecuencia de una palabra con la inicial en mayúsculas respecto de otras formas en un corpus, atributo que denominan "permanencia".

Delimitación

Un problema habitual en NER es la detección de dónde comienza y termina una entidad. La identificación de los límites de la entidad puede llevar a error cuando la entidad está compuesta de dos o más palabras (e.g. "John Fitzgerald Kennedy" y "Aeropuerto John Fitzgerald Kennedy") y sólo se detecta como entidad una parte, se detecta como varias entidades diferentes o bien se incluyen como parte de ella además otros términos que la rodean.

Palmer y Day [1997] proponen como estrategia para resolver este problema tomar la entidad de mayor número de palabras. Nadeau [2007] lleva a cabo esta estrategia, y se unen las entidades con los términos adyacentes que comiencen por mayúscula (y que se consideren entidades según la heurística de Mikheev citada anteriormente), y aquellos términos consecutivos identificados con la misma semántica.

Da Silva [2004] propone el uso de estadísticas computadas a partir de un corpus para determinar cuándo la presencia de un conjunto de palabras en minúsculas puede formar parte o no de una entidad.

Co-referencia

Las referencias en un texto a una entidad podrían considerarse tan valiosas como la propia entidad. Por ejemplo, en la frase "La casa de Ana, donde ella vive", "ella" hace referencia a Ana, que es una entidad de nombre. Dos o más elementos son co-referentes cuando se refieren a la misma situación descrita en el texto [Moens, 2006]. En general, el tipo de referencias que se trata de resolver son las fóricas, es decir, las que se refieren a información que se encuentra antes o después en el texto, y no fuera de él como ocurre con las referencias exofóricas. Las referencias fóricas sobre elementos que se encuentran antes en el texto son denominadas *anáforas*, como contrapartida a las *catáforas*, que se refieren a elementos que aparecen posteriormente en el texto.

La co-referencia en entidades ha sido ampliamente estudiada en la literatura y ya se introdujo como una de las tareas en MUC-6 y 7. Para su resolución se han aplicado diversos enfoques, desde la transformación a un problema de clasificación utilizando aprendizaje

automático [Mccarthy & Lehnert, 1995] [Ng & Cardie, 2002] a patrones gramaticales elaborados manualmente (Bontcheva, Dimitrov et al. 2002). A pesar de que los estudios teóricos describen la co-referencia como un complejo fenómeno lingüístico, afectado por múltiples factores, el estado del arte en cuanto a los enfoques estadísticos para su resolución muestra modelos basados en poco conocimiento (Uryupina 2007).

Alias

Los alias de una entidad son aquellas formas en las que puede escribirse esa entidad en un documento. Por ejemplo, para una entidad dada, "Francisco Pérez González", podemos tener como alias "Francisco Pérez", "Francisco P. González", "Paco", "Paco P. Glez.", "F. Pérez", etc.

En el trabajo de Cunningham et al. [1995] se usan 31 heurísticas para equiparar múltiples ocurrencias de nombres de compañías. Por ejemplo, dos entidades multi-término se consideran expresiones de una misma entidad cuando una es la secuencia inicial de la otra. Bontcheva et al. [2002] implementan en la herramienta GATE para extracción de información un módulo de reconocimiento de alias (además de otro para la co-referencia) basado en las anteriores heurísticas más otras adicionales. Estas heurísticas, algunas de ellas dependientes del tipo de entidad, tratan entidades compuestas, acrónimos y abreviaturas. Nadeau [2007] simplifica aún más esta labor mediante un algoritmo que denominan *Simple Alias Algorithm Resolution* y que consiste en agrupar entidades que comparten al menos una palabra de más de tres letras.

Li et al. [2004] tratan el problema de la resolución de co-referencia y alias entre documentos, comparando el uso de técnicas supervisadas y no supervisadas para esta tarea. Por otro lado, Poibeau [2006] pone de manifiesto la necesidad de utilizar etiquetado semántico.

Dependencia de dominio y género

El impacto del género documental (periodístico, científico, informal, etc.) y del tipo de dominio (jardinería, deportes, negocios, etc.) ha sido prácticamente ignorado en la

literatura sobre NER (Nadeau 2007). Salvando dominios como el de la biología, la literatura sobre trabajos en dominios específicos es escasa y la mayor parte de las herramientas NER o bien no son adaptables, o bien lo son a través de aprendizaje a partir de un corpus anotado. Una excepción es el caso de la herramienta ESpotter [Zhu et al., 2005]. En ella se asocia a cada entrada del léxico o patrón para el reconocimiento de entidades, una probabilidad de acierto para un dominio concreto, identificado con una URL o parte de ella. Estas probabilidades son estimadas utilizando la métrica *Google Conditional Probability* (Perkowitz, Philipose et al. 2004), dividiendo el número de páginas que contienen la entidad y su tipo (a modo de cadenas de texto) en un dominio concreto (un sitio web o parte de él), entre el número de páginas de ese dominio que contiene la entidad pero no su tipo.

En cuanto a la adaptación de herramientas NER a nuevos dominios, esto suele suponer una considerable pérdida de efectividad, como queda demostrado en el trabajo de Poibeau y Kosseim [2001]. En él se probaron algunos sistemas NER sobre la colección MUC-6, formada por noticias periodísticas, y sobre un corpus propietario constituido por transcripciones de conversaciones telefónicas y emails, y se observó una caída en el rendimiento de cada sistema de entre el 20% y el 40% de precision y recall. Las variaciones en las características tipográficas de las entidades, de su contexto o del estilo de escritura pueden provocar esta pérdida de eficacia. Moens [2006] reporta caídas dramáticas de la efectividad ante el procesamiento de informes policiales debido a las pocas destrezas en la escritura, unidas a la jerga utilizada y el uso de alias.

2.2. Técnicas de Reconocimiento de Entidades Nombradas

Las técnicas aplicadas en el reconocimiento de entidades han evolucionado desde la elaboración de patrones manuales a reglas obtenidas automáticamente mediante técnicas de aprendizaje.

Los sistemas de aprendizaje automático supervisado se basan en la creación de reglas a partir de ejemplos positivos y negativos anotados en un corpus. La idea general es que un experto anota los fragmentos de texto que servirán como un corpus de entrenamiento, y el sistema de aprendizaje producirá reglas a partir de estos ejemplos de modo que puedan ser aplicadas a instancias no conocidas por el sistema (Moens 2006).

Aunque la tendencia indica que los sistemas de aprendizaje automático producen mejores resultados que los basados en reglas manuales, las tasas de los últimos pueden ser muy elevadas: Jacobs y Rau [1990] consiguen clasificar artículos con un 92% de precision y un 88.5% de recall, mientras que Hayes y Weinstein [1990] alcanzan un 84% de precision y un 94% de recall en la clasificación de artículos sobre 675 categorías. El problema es que el trabajo para crear reglas tan afinadas es extremadamente costoso y de difícil mantenimiento si cambian los contenidos documentales o las categorías.

Por otro lado, también resulta costosa la anotación de corpus necesaria para los sistemas de aprendizaje automático. De hecho, este coste parece ser el mayor problema en el desarrollo de sistemas de extracción de información a gran escala o en su portabilidad a otros dominios (Moens 2006). Ante esta situación aparecen las técnicas de aprendizaje semi-supervisado, que no requiere grandes cantidades de datos anotados sino únicamente un reducido número de ejemplos. Finalmente, es posible eliminar por completo la necesidad de corpus anotados mediante técnicas de aprendizaje no supervisado, donde el sistema no cuenta con la guía del conocimiento del usuario y se establecen relaciones entre los datos según sus características.

2.2.1. Técnicas supervisadas

La aplicación de técnicas de aprendizaje supervisado a extracción de información se remonta a principios de los noventa, con los trabajos de Riloff y Lehnert [1993] entre otros. Los algoritmos empleados son muy variados y los más conocidos son los clasificadores y los ad-hoc.

Clasificadores

Los sistemas más utilizados son los llamados clasificadores, que utilizan métodos de clasificación con base probabilística, matemática, o incluso Inteligencia Artificial, como es el caso de las redes neuronales o los algoritmos genéticos. Uno de los clasificadores probabilísticos más conocidos son los Modelos Ocultos de Markov *(Hidden Markov Model* –HMM-), que han sido aplicados a NER desde los inicios (por ejemplo en Bikel et al [1997]). Otro ejemplo es el *Naive Bayes*, que asume independencia entre las características de un ejemplo etiquetado respecto a su clase y es muy utilizado en la clasificación de textos. Otros clasificadores probabilísticos utilizan distribuciones conocidas y se basan en el Principio de Entropía para la estimación de los parámetros. Este es el caso de los Modelos de Máxima Entropia *(Maximum Entropy Model* –ME-*)*, o el conocido como *Conditional Random Fields*, ambos utilizados en NER [Borthwick et al., 1998] .

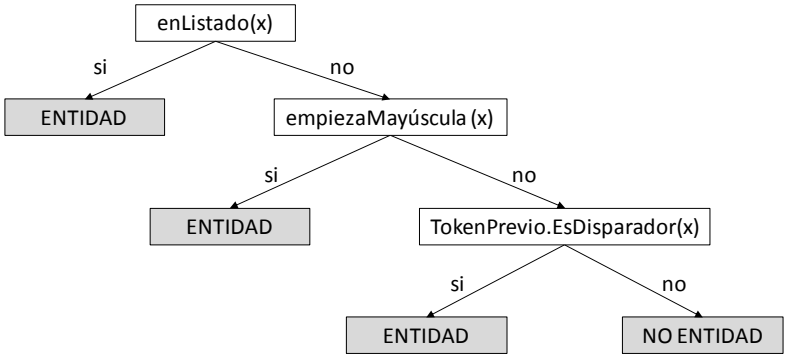


Figura 2. Ejemplo de árbol de decisión para detección de entidades.

En cuanto a los clasificadores matemáticos, el más representativo es la Máquina de Soporte Vectorial *(Support Vector Machines* –SVMs-*).* Son clasificadores basados en espacios vectoriales, donde el objetivo es encontrar una función capaz de separar las clases de manera que esté lo más alejada posible de cualquier punto en el conjunto de entrenamiento (Manning, Raghavan et al. 2008). Aplicaciones de esta técnica en NER podemos encontrarlas por ejemplo en Asahara y Matsumoto [2003].

También son muy utilizados los clasificadores donde el conocimiento es representado en forma de descripciones simbólicas, por ejemplo reglas o jerarquías de conceptos [Chen, 1995]. Este es el caso por ejemplo de la representación como árboles de decisión *(Tree Learning)*. Estos últimos consisten en la construcción de árboles de decisión de tal manera

que siguiendo la ruta de la raíz a las hojas del árbol sea posible determinar la clase a la que pertenece una observación dada. El atributo que más los diferencie se usa en la regla de decisión del primer nodo y así sucesivamente con los nodos inferiores (Figura 2). Ejemplos de trabajos en NER con árboles de decisión se encuentran en Sekine y Grishman [1998].

Sistemas Ad-Hoc

A diferencia de los clasificadores, algunos sistemas generan reglas de un modo más *ad-hoc* mediante el uso de ciertas características de las entidades o del texto que las contiene. La ventaja es que el número de ejemplos necesarios para entrenar suele ser inferior al requerido en los clasificadores. Es el caso de muchos de los sistemas de extracción de información utilizados en las conferencias MUC, que construían automáticamente reglas a partir de un conjunto de ejemplos generalmente positivos. Ejemplos de este tipo de sistemas son *AutoSlog* [Riloff & Jones, 1999] o RAPIER [Califf & Mooney, 2004]. Algunos de ellos, como es el caso de RAPIER, son directamente aplicables a NER, puesto que permiten la captura de entidades (sistemas *single-slot),* frente a aquellos que requieren la construcción de escenarios a partir de estas entidades (sistemas *multi-slot)*.

Los métodos de generación de reglas en este tipo de sistemas se pueden clasificar en métodos de compresión o métodos de cobertura *(covering)* [Califf & Mooney, 2004]. Los algoritmos de compresión tratan en cada iteración de comprimir las reglas construidas reemplazándolas por otras. El proceso finaliza cuando no se encuentran reglas que permitan ser comprimidas. Por otro lado en cobertura los ejemplos positivos cubiertos por las reglas generadas en cada iteración son eliminados, y los ejemplos restantes son utilizados para aprender nuevas reglas. El proceso finaliza cuando no existen ejemplos positivos sin cubrir por ninguna regla. Estos últimos sistemas tienden a ser más eficientes, pero menos exhaustivos que los de compresión [Califf & Mooney, 2004].

Por otro lado, según la generación de reglas vaya de lo más general a lo más específico o a la inversa, podemos clasificar los algoritmos en top-down o bottom-up respectivamente. En el primer caso los sistemas parten de reglas muy generales (típicamente reglas que cubren todos los ejemplos positivos) y tratan de especializarlas evitando que reconozcan

ejemplos negativos pero manteniendo el reconocimiento de los positivos. En el caso de los sistemas bottom-up por el contrario, se parte de reglas muy específicas (habitualmente una por cada ejemplo positivo) que son generalizadas para cubrir otros ejemplos positivos al mismo tiempo. Estos últimos sistemas tienden a ser más precisos, pero pueden crear reglas demasiado específicas y caer en el *overfitting*, con malos resultados cuando se aplican a corpus diferentes. Por otro lado, los sistemas top-down pueden resultar muy ineficientes si el espacio de búsqueda es muy amplio (muchos atributos y posibles valores de esos atributos) [Califf & Mooney, 2004].

Un ejemplo de sistema NER de tipo cobertura y generación de reglas top-down podemos encontrarlo en Wu y Pottenger [2005]. Otros ejemplos, en este caso de sistemas de extracción de información aplicables a NER, son Whisk [Soderland, 1999], Amilcare [Ciravegna & Wilks, 2003a] y el ya citado RAPIER. Whisk es un algoritmo de tipo *covering* que genera reglas a partir de semillas iniciales en un modo top-down. RAPIER genera reglas en modo bottom-up utilizando el método de compresión, es decir, trata de generalizar las reglas creadas, sustituyendo a otras más específicas. Por último Amilcare utiliza un algoritmo llamado (LP)2 (*Learning Pattern by Language Processing*), también de aprendizaje bottom-up de reglas, pero en este caso de tipo *covering* [Ciravegna et al., 2001].

Co-entrenamiento

Cualquiera de las técnicas mencionadas anteriormente puede aplicarse para generar varios modelos que se combinen entre sí. Esta aproximación es conocida como co-entrenamiento *(co-training),* y consiste en que se entrenan varios modelos a partir de un corpus anotado, pero utilizando diferentes clasificadores y/o conjuntos de atributos. Con los modelos generados se anota un corpus no etiquetado, de manera que los elementos clasificados por los diversos clasificadores con mayor confianza se añaden al corpus de entrenamiento y se repite el proceso. Ejemplos de la aplicación de esta técnica para el reconocimiento de entidades los encontramos en los trabajos de Collins y Singer [1999] y Kozareva et al. [2005].

2.2.2. Técnicas semi-supervisadas

Las técnicas semi-supervisadas, también conocidas como técnicas débilmente supervisadas (*Weakly Supervised Techniques)*, consisten en utilizar datos anotados combinados con los no anotados para construir los clasificadores. Esta aproximación reduce la cantidad necesaria de etiquetado de corpus, disminuyendo así uno de los grandes problemas del aprendizaje supervisado. Como contrapartida, debe proveerse de corpus no anotados que traten de mejorar, generalmente de forma progresiva, el clasificador creado con el corpus inicial anotado (que puede consistir únicamente en una serie de patrones o ejemplos ya etiquetados). Ji y Grishman [2006], afirman que el uso de grandes colecciones de documentos no es suficiente por sí misma, sino la selección de información rica en nombres propios y co-referencias.

Aprendizaje Activo

Es posible incorporar ejemplos etiquetados a lo largo de todo el proceso de aprendizaje mediante una técnica conocida como Aprendizaje Activo *(Active Learning* -AL*)*, donde es el propio sistema el que proporciona al usuario los candidatos para que los corrija o etiquete, de modo que puedan ser utilizados como nuevas semillas para reentrenar el modelo. Existen diversas técnicas para la selección de los ejemplos a etiquetar [Settles, 2010]:

- *Uncertainty sampling*: se seleccionan las instancias con menos certeza de que sean válidas.
- *Query by Committee*: cuando el aprendizaje se realiza con varios algoritmos a la vez, se seleccionan las instancias donde más desacuerdo exista entre ellos.
- *Density-Weighted Methods*: se seleccionan las instancias no sólo con menos certeza de validez, sino que además son representativas de la distribución de entrada. De este modo se trata de evitar la selección de *outliers* a la que son propensos los métodos anteriores.
- *Expected Model Change*: se seleccionan aquellas instancias que más puedan influir en el modelo.
- *Variance Reduction and Fisher Information Ratio*: se eligen las instancias bajo el criterio general de minimizar la varianza.

- *Estimated Error Reduction*: se seleccionan las instancias que minimizan el error esperado.

Son diversos los trabajos en NER que utilizan esta técnica. Por ejemplo, se ha probado con éxito para trasladar sistemas de biomedicina (corpus GENIA) al dominio de la radio-astronomía (*abstracts* de *The Astronomy Bootstrapping Corpus*, de la NASA). La aproximación combina Modelos Condicionales de Markov (CMM) con la anotación por parte del usuario de aquellos ejemplos con mayor desacuerdo entre los modelos (estrategia *Query By Committee)* [Hachey et al., 2005]. Shen et al. [2004] aplica AL a NER mediante el uso de Máquinas de Soporte Vectorial. En este caso se investigan los efectos de la estrategia a utilizar al seleccionar los ejemplos a etiquetar, con métodos muy específicos de los algoritmos de este tipo. También se aplica SVM en el trabajo de Vlachos [2008] y en el de Li et al. [2008]. En el primero se analizan los criterios de parada mientras que en el segundo se implementan mejoras sobre el algoritmo SVM. En ambos casos se utiliza la estrategia u*ncertainty sampling* para seleccionar los ejemplos a etiquetar. En Tomanek et al. [2007] también se analiza el uso de AL para la anotación de corpus en NER, usando para ello modelos de Máxima Entropía.

También podemos encontrar técnicas de aprendizaje activo en sistemas ad-hoc. Es el caso de Whisk [Soderland, 1999] y RAPIER [Califf & Mooney, 2004]. En Whisk, los ejemplos mostrados al usuario para etiquetar pertenecen a tres grupos de ejemplos: aquellos ya cubiertos por alguna regla (para mejorar la precisión), aquellos que "casi" encajan en una regla (para mejorar la exhaustividad), y ejemplos no cubiertos por ninguna regla. En cuanto a RAPIER, se incorporó el aprendizaje activo al algoritmo original mediante la estrategia de *uncertainty sampling* [Thompson et al., 1999]. Wu y Pottenger [2005] también incorporan en su metodología aprendizaje activo, sugiriendo al usuario nuevos segmentos de texto a etiquetar a partir del contexto de otros ejemplos positivos. Por último, *Amilcare* [Ciravegna & Wilks, 2003a] reentrena con las correcciones aportadas por el usuario ante los documentos anotados por la herramienta.

Bootstrapping

Una de las técnicas más conocidas de aprendizaje semi-supervisado (a veces considerada en la literatura como una técnica de aprendizaje no supervisado) es la denominada bootstrapping, llamada también Expansión o *Self Training* en la bibliografía [Moens, 2006]. Consiste en suministrar ejemplos etiquetados o patrones como semillas, a partir de los cuales comienza el proceso de aprendizaje de modo automático. En general la idea subyacente es que, una vez tenemos el clasificador inicial, es posible encontrar en el corpus no anotado nuevos elementos que clasificar gracias a un contexto conocido, o bien a la inversa, nuevos contextos que aprender gracias a elementos que ya conoce. Habrá contextos que siempre aparezcan por ejemplo seguidos por un nombre de persona en el corpus anotado (e.g. Sr.). Si encontramos un nuevo token *T* en este contexto en el corpus no anotado, podemos saber con bastante certeza que se trata de un nombre de persona. Si el clasificador puede aprender que *T* es un nombre de persona, puede etiquetarla correctamente sin ningún contexto indicativo (aunque obviamente puede errar si es un término ambiguo), e incorporar a las reglas otros contextos de *T* que aparezcan de forma consistente.

Algunas de las asunciones de esta técnica son que las semillas están presentes en el corpus no etiquetado y que existe cierta redundancia en los contextos, que es además no ambigua respecto a la categoría semántica de aquello que se va a recuperar [Jones, 2005]. Adicionalmente, un problema común al bootstrapping y al co-entrenamiento es la medida de la relevancia de la información extraída necesaria para incorporarla como válida. Esto es especialmente importante, ya que los resultados de este tipo de sistemas empeoran rápidamente si no se realiza una adecuada selección de las nuevas semillas que se van incorporando al proceso. En general esta medida además determina el criterio de parada del algoritmo (cuando ya no sea posible localizar más información relevante).

La técnica de bootstrapping ya es descrita, aunque no implementada de modo automático, por Hearst [1992]. Hearst extrae de modo manual patrones léxicos para la identificación de relaciones de jerarquía entre términos, y utiliza estos términos ya relacionados para capturar el contexto y modelar los elementos comunes, formando de este

modo nuevos patrones. Yarowsky [1995] aplica esta técnica de modo automático a técnicas de desambiguación y Brin [1999] la aplica con el objetivo de construir expresiones regulares capaces de detectar pares {título, autor}, como por ejemplo {Isaac Asimov, *The Robots of Dawn*}. Este algoritmo fue pensado para su aplicación sobre la Web y muestra que ya desde los 90 era utilizada como corpus no etiquetado para el aprendizaje.

En el trabajo de Riloff y Jones [1999] se aplica la técnica de bootstrapping a NER y se define el bootstrapping mutuo *(mutual bootstrapping).* Esta técnica consiste en utilizar como semillas no sólo las entidades sino también el contexto en el que se encuentran a modo de patrones, de modo que se genera al mismo tiempo un conjunto de elementos y un diccionario de patrones (Figura 3). Su funcionamiento es el siguiente. Se seleccionan un conjunto de entidades de un tipo dado (e.g. Bolivia, Guatemala, Honduras, con el tipo "país"), y se extraen los patrones encontrados en torno a estas entidades en el corpus (e.g. "oficinas en X", "instalaciones en X"). Se selecciona el mejor de estos patrones según las entidades capturadas (las que se conocen) y se incorporan las que se consideran las mejores entidades que captura (las que más veces son capturadas por los patrones extraídos; este proceso es denominado meta-bootstrapping). Se repite el proceso utilizando como semillas las anteriores más las nuevas potenciales entidades incorporadas y se detiene cuando el mejor de los patrones obtenidos se encuentra por debajo de un umbral. Los resultados de logrados son de entre un 18-54% de recall y 45-74% de precision para tres tipos de entidades (compañías, localizaciones y títulos de personas).

Este trabajo resulta ser de gran influencia para posteriores investigaciones, como el amplio trabajo de Jones [2005] sobre esta técnica aplicada a nombres de persona, organismos y localizaciones, incorporando además aprendizaje activo. También Cucchiarelli y Velardi [2001] utilizan la misma técnica pero con algunas variaciones, entre ellas la de utilizar como semillas la salida de sistemas NER existentes (que denominan clasificador temprano –*early NE classifier*-) en lugar de entidades anotadas manualmente.

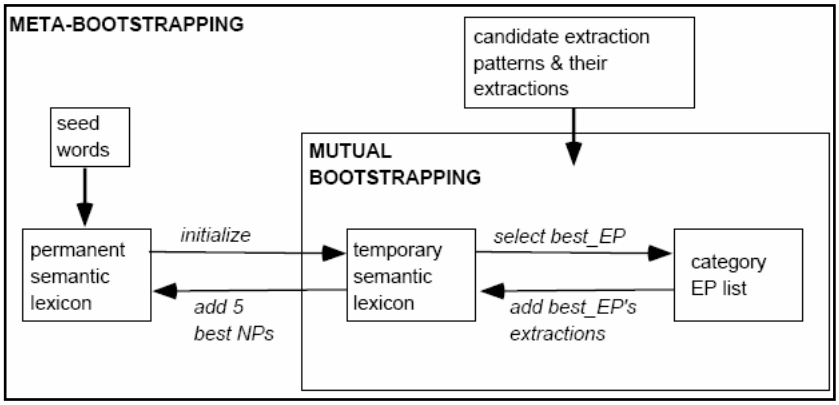


Figura 3. Procesos de meta y mutual bootstrapping (Riloff, Jones 1999).

Otras aplicaciones de bootstrapping a NER las podemos encontrar en Collins y Singer [1999] y en Pasca [2007]. Los primeros aplican un algoritmo muy similar al de Yarowsky en desambiguación para inducir reglas de contenido y contexto sobre ciertos tipos de entidades (Tabla 2). Pasca aplica bootstrapping para la identificación de entidades a partir de las consultas de los usuarios sobre la Web.

Tabla 2. Patrones iniciales de contenido como semillas para aprendizaje en NER (Collins, Singer 1999).

Full-string = New York	Location
Full-string = California	Location
Full-string = U.S.	Location
Contains (Mr.)	Person
Contains (Incorporated)	Organization
Full-string = Microsoft	Organization
Full-string = IBM	Organization

2.2.3. Técnicas no supervisadas

La ausencia de información proporcionada manualmente, ya sea mediante el etiquetado de corpus o la incorporación de semillas, suele considerarse aprendizaje no supervisado. Sin embargo, "estas distinciones no siempre son claras, ya que en algunos sistemas considerados como no supervisados puede argumentarse que el conocimiento necesario para generar datos etiquetados ha sido sustituido por otros recursos de conocimiento,

como las heurísticas u otro tipo de conocimiento inicial incorporado al sistema"[4] [Nadeau et al., 2006].

El *clustering* es una de las técnicas no supervisadas más conocidas, donde el objetivo es encontrar similitudes entre elementos y agruparlos en conjuntos cuyo número puede o no venir establecido a priori. Esta técnica ha sido aplicada en recuperación de información, tanto para la realimentación de consultas mediante la identificación de información similar a un documento seleccionado por el usuario, como para la mejora de la visualización de los resultados mediante su agrupación por categorías. Sin embargo, ha tenido escaso éxito en el reconocimiento de entidades propiamente. Uno de los trabajos donde se aplica es el realizado por Lin y Pantel [2001], donde se añade la información sobre la dependencia sintáctica entre palabras para agrupar aquellas semánticamente similares. Con esta técnica muestran cómo generar clases semánticas (e.g. marcas de coches, tipos de medicamentos, provincias), aunque no es posible determinar de modo automático la etiqueta de la clase.

Otras técnicas consideradas no supervisadas recurren al uso de recursos anotados semánticamente, como los listados de términos o las ontologías. Algunos trabajos incluso parten de patrones o entidades conocidas, pero al no incorporar nuevo conocimiento etiquetado al sistema en forma de nuevos patrones o entidades, son consideradas por los propios autores como no supervisadas en lugar de semi-supervisadas. Este es el caso por ejemplo de los trabajos de Etzioni con su sistema de extracción de información KnowItAll [Etzioni et al., 2005] o el trabajo de Nadeau con su sistema de reconocimiento de entidades YooName (Nadeau 2007).

El sistema KnowItAll tiene como objetivo la extracción de información de la Web mediante el uso de patrones independientes de dominio inspirados por Hearst, unidos a predicados n-arios que constan de palabras válidas como argumentos del elemento o la relación que describen (Tabla 3). Esta información, unida a restricciones de tipo morfo-sintáctico, se combina para generar reglas (Tabla 4), que serán lanzadas a la Web para

[4] *[...] the distinction between supervised and unsupervised systems is not always clear. In some systems that are apparently unsupervised, it could be argued that the human labour of generating labeled training data has merely been shifted to embedding clever rules and heuristics in the system.*

localizar cadenas de texto que encajen. Para determinar si la información recuperada es correcta para un predicado dado (vamos a suponer que es una instancia de una clase concreta, que es el caso si el predicado es unario), se utiliza una variante del algoritmo PMI-IR (*Pointwise Mutual Information-Information Retrieval)* introducido por Turney [2001]. El umbral para cada predicado se determina entrenando un clasificador *Naive Bayes* con diez ejemplos positivos y otros diez negativos para cada predicado.

Tabla 3**.** Ejemplo de predicados unarios y binarios, y patrones del sistema *KnowItAll.*

Predicado: City	Patrón: NP "and other" <clase1>
Etiquetas: "city", "town"	Patrón: NP "or other" <clase1>
Predicado: Country	Patrón: <clase1> "especially" NPList
Etiquetas: "country", "nation"	Patrón: <clase1> "including" NPList
Predicado: capitalOf (City,Country)	Patrón: <clase1> "such as" NPList
Etiquetas: "capital of"	Patrón: "such" <clase1> "as" NPList
Etiquetas clase-1: "city", "town"	Patrón: <clase1> "is the" <relation> <clase2>
Etiquetas clase-2: "country", "nation"	Patrón: <clase1> "," <relation> <clase2>

KnowItAll puede ser utilizado como un sistema de bootstrapping. De hecho, Etzioni utiliza parte de su sistema de extracción de información KnowItAll para la identificación de entidades de categorías predefinidas mediante esta técnica. Posteriormente, Nadeau [2007] mejora este algoritmo tanto en lo que se refiere a la generación de los wrapper capaces de extraer el máximo de entidades de cada web, como a su filtrado, cuestiones que trata de resolver mediante heurísticas y aprendizaje automático. Otras aplicaciones de la herramienta incluyen la identificación de jerarquías de clases mediante la modificación de los patrones para la incorporación de relaciones entre nombres comunes (se asume que son nombres comunes simplemente cuando aparecen en minúsculas). La evaluación de las clases obtenidas se realiza utilizando WordNet (ver Sección 2.3.2).

WordNet es utilizada también por Moldovan y Girju [2003] para identificar elementos relacionados y a partir de ellos extraer de la Web patrones en los que aparezcan con cierta frecuencia. Este trabajo amplía el número de patrones aportados por Hearst (por ejemplo, en lo que respecta a los patrones para detectar relaciones parte-todo), lo que puede ayudar

en las tareas de NER. Por otro lado, Alfonseca y Manandhar [2002a] hacen uso de WordNet para la clasificación de entidades. Parten de un algoritmo utilizado para la población de WordNet y descrito en Agirre et al. [2000] que consiste en determinar los términos que co-ocurren con un *synset* de WordNet y anotar sus frecuencias a partir de la Web. Esta información es comparada con la información de contexto de un término en el corpus, de modo que se asigna el concepto al *synset* más similar.

Tabla 4. Regla generada a partir de la combinación del patrón NP1 "such as" NPList2, y el predicado City. En realidad en el sistema estas reglas son representadas mediante una gramática en notación BNF.

Predicado	City
Patrón	NP1 "such as" NPList2
Restricciones	head(NP1)= "cities"
Relaciónes	City(head(each(NPList2)))
Palabras clave	"cities such as"

Otra técnica que podría considerarse no supervisada es la aplicada por Shinyama y Sekine [2004a]. Ellos observaron que existía una fuerte correlación entre ser una entidad y aparecer en un momento determinado en diferentes fuentes de noticias periodísticas y utilizaron esta información para la captura de entidades.

2.3. Atributos y recursos

2.3.1. Atributos

El reconocimiento de entidades, tanto con métodos supervisados como con poca o ninguna supervisión, requiere el previo modelado de la información para que pueda ser procesada automáticamente. En el caso del aprendizaje automático supervisado, cada elemento a reconocer viene modelado como un vector de características o atributos que describen a ese elemento. Típicamente cada dato es representado por uno o varios atributos booleanos, numéricos o nominales. Por ejemplo, podríamos tener cada palabra de un texto representada de la siguiente manera:

- Un atributo booleano que indica si la palabra está en mayúsculas o no.
- Un atributo numérico que indica la longitud de la palabra en caracteres.
- Un atributo nominal que determina qué tipo de entidad es (por ejemplo: persona, organismo, localización, miscelánea, no_entidad).

En un algoritmo de aprendizaje primero se extraería, para cada palabra de un corpus etiquetado, la información de estos atributos, que será la que la represente. Habitualmente el último atributo es aquel que queremos que el algoritmo aprenda, es decir, que dados los restantes atributos el modelo de aprendizaje pudiera determinar ese último atributo, llamado habitualmente "atributo de clase". Siguiendo el ejemplo anterior, el modelo construido nos indicaría a qué tipo de entidad pertenece una palabra que está en mayúsculas y tiene tres letras, por ejemplo.

Los atributos utilizados dependen de la información que queremos predecir, es decir, dependen del atributo de clase. En reconocimiento de entidades podemos encontrar tanto sistemas que llevan a cabo por separado las tareas de identificación y clasificación, como sistemas que realizan ambas tareas al mismo tiempo. Los primeros son menos habituales. algunos ejemplos son las herramientas NERUA [Ferrández et al., 2005](Tabla 5) y Freeling (Atserias, Casas et al. 2006).

La bondad de las predicciones de un modelo de aprendizaje dependerá de diversos factores. Algunos de esos factores son: el algoritmo de aprendizaje, la existencia de un número similar de elementos para cada uno de los valores del atributo de clase (la muestra debe ser balanceada para obtener modelos válidos) y la representatividad y cantidad de información que aporte el corpus para los datos que pretendemos aprender. Pero Curran y Clark [2003] demuestran que además el conjunto de atributos puede tener una fuerte influencia en los resultados. De hecho, la ingeniería de atributos (*Feature Engineering)* se ocupa de determinar cuáles son los atributos más adecuados a los datos para un algoritmo, de modo que resulte en un aprendizaje más efectivo.

También en técnicas no supervisadas es habitual el modelado de la información mediante vectores de atributos que la describan. Por ejemplo, en las técnicas de clustering la agrupación en clases puede realizarse según la similitud de los vectores de atributos. Por

otro lado, las técnicas semi-supervisadas suelen modelar mediante patrones (habitualmente expresiones regulares) la combinación de tales atributos.

Tabla 5. Atributos de identificación de entidades en el sistema NERUA [Ferrández et al., 2005].

Atributo	Definición
e	la palabra a ser clasificada
cntxt[1-6]	palabra del contexto en posición ±1, ±2, ±3
CNTXT[1-7]	palabra en mayúsculas en la posición 0, ±1, ±2, ±3
EE[1-3]	palabra +1,+2,+3 en diccionario entre entidades
pos	posición de la palabra en la frase
eMy	palabra completa en mayúsculas
eDic	palabra en algún diccionario
eDisp	palabra en diccionarios de disparadores
cntxtDisp	palabra en la posición ±1, ±2, ±3 en diccionarios de disparadores
eLem	lema de la palabra
eRaiz	raíz de la palabra
SubStr[1-5]	±2, ±3 y la mitad de caracteres de la palabra

Siguiendo la clasificación dada por Nadeau y Sekine [2007], podemos agrupar los tipos de atributos en cuatro categorías: léxicos, sintácticos, semánticos y de discurso. Categorías que a su vez corresponden con los niveles del lenguaje y que representan el nivel al que puede ser descrita una unidad de texto. Estas unidades pueden ser una única palabra o un conjunto de ellas, formando una palabra compuesta o incluso un sintagma (para facilitar la lectura en los siguientes aparatados, nos referiremos a ellas en todos los casos como "palabras").

Atributos léxicos

Los atributos léxicos hacen referencia a cuestiones que afectan a una unidad por sí misma, independientemente de la frase o texto en el que se encuentra. Tratan fundamentalmente aspectos tipográficos y morfológicos de la palabra. Algunos de ellos son:

- Mayúsculas: si las palabras incorporan letras en mayúsculas, ya sea en la inicial, en su totalidad o parcialmente.

- Tipo de caracteres: si las palabras contienen símbolos de puntuación (e.g. puntos, apóstrofes, guiones, comillas), dígitos de cualquier tipo (e.g. cardinales, ordinales, caracteres romanos) o símbolos (e.g. arroba, *ampersand*). Este tipo de atributo depende del tipo de tokenización que se haya realizado, ya que en muchas ocasiones algunos de estos tipos de caracteres aparecen como un token independiente (por ejemplo, el apóstrofe es considerado habitualmente en los parsers en inglés como un token independiente, con lo que la frase *Tom's house* sería dividida en tres tokens: *Tom 's house*). En estos casos algunos de estos atributos no se analizan como parte de una palabra, sino en todo caso como tipo de palabra (token) contextual a una dada (e.g. si una palabra está seguida o precedida de otra que es un número, un apóstrofe, un punto, etc.).
- Categoría morfológica: son habituales los atributos que indican la categoría de la palabra (sustantivo, adjetivo, preposición, etc.), la forma normal de la misma, su lexema, raíz, y los prefijos y sufijos que pueda contener.
- Otras: existen otros atributos que pueden utilizarse a nivel de léxico, como pueden ser la longitud de la palabra, o incluso la propia palabra en sí. Si se toma como atributo la propia palabra suelen escogerse únicamente aquellas más frecuentes para no aumentar demasiado la dimensionalidad de los vectores, y con ella el tiempo de cálculo.

Atributos morfosintácticos

Los atributos sintácticos indican la función que desempeña la palabra en la frase. La anotación morfosintáctica es habitualmente conocida con el término *Part of Speech Tagging (PoS Tagging* –POS-*).* En algunos casos se utilizan analizadores que extraen toda la información morfosintáctica, pero en otros casos es suficiente con utilizar los denominados *shallow parsers*, que realizan un análisis más superficial, extrayendo los sintagmas que componen las oraciones. La dependencia sintáctica con las palabras del contexto es otro de los atributos que suele utilizarse tanto en el área de desambiguación semántica como en reconocimiento de entidades, como ocurre en el trabajo de Lin y Pantel [2001] por ejemplo.

Atributos semánticos

Los atributos semánticos hacen referencia al significado de la palabra. Generalmente se utilizan recursos externos para determinar este tipo de información, como los diccionarios o las ontologías, que pueden además aportar información sobre la relación semántica de esa palabra con otras (e.g. sinonimia, homonimia, hiperonimia, hiponimia, meronimia). Un atributo frecuentemente utilizado es aquel que indica la inclusión de la palabra o palabras, total o parcialmente, en alguno de estos recursos.

Atributos de discurso

Los atributos de discurso hacen referencia a aquellas características de la palabra que dependen de un contexto superior al establecido por la frase. Algunos de los atributos que podemos encontrar son los siguientes:

- Ocurrencia de otros elementos similares en el texto, sus características, localización en la frase o el documento, y frecuencia. Un ejemplo es la heurística aplicada por Nadeau [2007], en la que analiza si existen otras palabras iguales escritas en mayúsculas o minúsculas para determinar si se trata de una entidad o no.
- Meta-información: información asociada a la estructura del propio documento (por ejemplo, código HTML que contiene a un elemento, o sección XML) o referente a información general del mismo (por ejemplo, URI, cabecera, etc.).
- Memoria: etiqueta más recientemente utilizada para la misma palabra en el corpus. Este atributo es utilizado inicialmente por Malouf [2002] como ayuda en la identificación de nombres de persona que frecuentemente son citados la primera vez con sus apellidos, y las siguientes veces sin ellos, dificultando su reconocimiento.

Además, es frecuente encontrar atributos referidos a las palabras en una ventana de uno, dos o más palabras alrededor, que pueden indicar cualquiera de las características descritas anteriormente. Es el caso por ejemplo de los disparadores (*triggerwords)*, que anteceden o preceden una entidad (e.g. “Sr.” para entidades de tipo persona).

2.3.2. Recursos

Las técnicas de reconocimiento de entidades suelen apoyarse en el uso de recursos semánticos como los diccionarios, las ontologías, patrones y *templates* que determinan la semántica asociada a una unidad de texto. Algunos estudios demuestran que el uso de recursos de este tipo contribuye notablemente a la eficacia de los sistemas de extracción de información, y que su influencia es mayor que las decisiones de diseño relativas al algoritmo utilizado (Iria, Ireson et al. 2006)(Carreras, Marquez et al. 2002). Otros estudios, sin embargo, ponen de manifiesto que no es así [Malouf, 2002]. En cualquier caso, se trata de un aspecto difícil de medir, ya que depende de la cantidad de información relevante almacenada en los recursos y del uso que se dé a la misma, influido a su vez por el modo en que se incorpora a los algoritmos de reconocimiento de entidades. Algunos recursos contienen diferentes tipos de entidades, previamente clasificadas, que pueden utilizarse para identificarlas en los textos, ya sea directamente o mediante su incorporación como atributos indicativos de la posible existencia de una entidad. Este es el caso de los listados y de recursos como Yago y Wikipedia. Otros recursos, como WordNet y FrameNet, son utilizados para extraer atributos léxicos de las propias entidades o de su contexto.

Listados

Los listados de palabras son habitualmente utilizados en reconocimiento de entidades (denominados generalmente *gazetteers*). Podemos encontrar listados de palabras comunes o diccionarios (útiles para la desambiguación entidad-nombre común), de categorías de entidades (listados de ciudades, por ejemplo), o también de disparadores.

Existen diversas estrategias para determinar si una palabra (simple o compuesta) se encuentra en un listado o no. Por ejemplo, puede considerarse válida únicamente si la palabra es exactamente igual a la del listado. En otros casos se aplican algoritmos de *fuzzy matching*, tratando de emparejar términos a pesar de no ser exactamente iguales. La técnica de *Soundex* [Raghavan & Allan, 2004] trata de localizar palabras que, aunque no aparecen escritas igual, tienen un sonido similar, lo que resulta muy útil si los datos provienen de grabaciones y están sujetos a errores. Otra técnica empleada es la distancia de edición, que

se traduce en el número de cambios que es necesario realizar a una palabra para transformarla en otra. Puede considerarse que una palabra es igual a otra si el número de cambios es menor a un determinado umbral. Son diversos los algoritmos que se emplean para calcular esta distancia. Algunos de los más conocidos son el algoritmo de Levenshtein [Levenshtein, 1966], y el de Jaro-Winkler [Winkler, 1999]. Este último permite dar más peso a la coincidencia en los primeros caracteres, lo que puede resultar útil en ciertos nombres propios.

Otro problema a la hora de reconocer palabras de una lista son las palabras compuestas. Son útiles en esta tarea las máquinas de estados finitas (FSM), que aceleran la comparación de las palabras en un texto con los listados implementados en forma de autómatas finitos. Otro algoritmo es el que se utiliza en la herramienta GATE [Cunningham et al., 2013], basado en la idea de Atanas Kiryakov de que utilizar tablas hash resulta más eficiente que utilizar máquinas de estados. El algoritmo empleado en GATE resulta de media más eficiente en tiempo y consumo de memoria, y consiste en utilizar tantas tablas hash como número de tokens componga la palabra compuesta más larga del listado. Estos tokens se descomponen y se almacenan en la tabla hash correspondiente según su longitud (primer token, primer token + segundo, etc.). La búsqueda entonces se realiza en la tabla apropiada según la longitud de la palabra o frase candidata a estar en los listados.

WordNet

WordNet [Princeton University, 2010] es una base de datos léxica del inglés elaborada y mantenida por el Departamento de Ciencia Cognitiva de la Universidad de Princeton. Contiene alrededor de 150.000 palabras, donde cada una de ellas es asociada a un concepto. Estos conceptos son representados mediante agrupaciones de palabras que tienen el mismo significado (sinónimos) y hay en torno a 115.000 (aproximadamente el 17% de las palabras son polisémicas y alrededor de un 40% tiene uno o más sinónimos). A estos conjuntos de palabras se les denomina *synsets*, y cada uno de ellos cuenta con definiciones, indicación de frecuencia de cada acepción para cada palabra (determinado en base a análisis mediante corpus) y relaciones léxico-semánticas con otros *synsets.* WordNet únicamente recoge clases

abiertas de palabras, es decir, sustantivos, adjetivos, adverbios y verbos. Las relaciones dependen de la categoría, de modo que, además de la sinonimia dada por los propios *synsets,* para los sustantivos podemos encontrar relaciones de antonimia, hiponimia (subordinación) y meronimia (parte-todo), para los verbos tenemos antonimia, troponimia (tipo de hiponimia para verbos) e implicación (*entailment),* y para los adjetivos y adverbios existen relaciones de antonimia.

La red semántica resultante es útil no sólo para su consulta manual a modo de diccionario, sino también para su uso en sistemas que requieran procesamiento del lenguaje natural. Para ello WordNet puede ser descargada bajo licencia BSD (*Berkeley Software Distribution)* y provee no sólo de interfaces para navegar por ella vía Web, sino de descargas en diversos formatos y API's en diversos lenguajes para su consulta.

Son numerosos los proyectos y aplicaciones que se han creado sobre WordNet[5]. Uno de los más relevantes es *EuroWordNet*, proyecto de la Unión Europea con el objetivo de crear una base de datos léxica similar para algunos de los idiomas de la Unión Europea, y conectarlos entre sí. Aunque se establecieron especificaciones, no se llegó a este objetivo y actualmente existe una asociación con este propósito (*Global WordNet Association)* y diversos proyectos en los que se realizan tareas similares (por ejemplo, *MultiWordNet*, que relaciona una versión de WordNet en italiano con WordNet en inglés).

FrameNet

FrameNet (Fillmore, Baker C.F. 2008) es una base de datos de frames semánticos desarrollada y mantenida por el Instituto de Ciencias de la Computación de la Universidad de Berkeley. Contiene más de 800 frames, que son representaciones esquemáticas de un objeto, estado o evento, con el objetivo de documentar el uso de cada palabra en cada uno de sus sentidos. Cada frame (por ejemplo, *Being_born)* contiene una descripción del concepto, ejemplos de frases con ese concepto (e.g. *She was born in 1978*), los roles (e.g. tiempo, lugar) y colocación de los mismos que constituyen esas frases (valencias*)*. Además, se indica la frecuencia con la que ocurren esas valencias.

[5] Pueden consultarse en http://wordnet.princeton.edu/wordnet/related-projects/

Como ocurre con WordNet, puede consultarse en línea o descargarse en diversos formatos (incluso OWL) de forma gratuita para propósitos de investigación. Actualmente existen diversos proyectos[6] sobre FrameNet, entre ellos su elaboración en diversos idiomas (ya hay versiones en español, alemán y japonés).

Yago

Yago es una base de datos con entidades y relaciones elaborada en el Instituto Max Planck [Suchanek et al., 2009] que forma parte del proyecto YAGO-NAGA, surgido en 2006 con la finalidad de crear una base de datos precisa de *facts* comunes (relaciones entre instancias) que sea accesible computacionalmente. Se trata de una ontología elaborada en RDFS que contiene como instancias las entradas de Wikipedia y como clases las de WordNet. Para ello son utilizadas las categorías asociadas a las entradas en Wikipedia, de modo que por medio de heurísticas se le asocia el *synset* de WordNet más parecido. En la ontología entonces se crea una relación de tipo en el que el título de la entrada en Wikipedia pasa a ser la instancia y el *synset* de WordNet asociado, la clase. También mediante heurísticas aplicadas a las categorías de una entrada se establecen relaciones entre instancias, resultando los llamados *facts* (ej. *Born_In, Familiy_name_of, during, Has_area, etc.)*. La ontología ha sido creada en su mayor parte de un modo automático, y presenta entre un 90 y un 98% de acierto en función del tipo de relación y según los test manuales realizados mediante muestreos [Suchanek et al., 2007]. Actualmente cuenta con más de 10 millones de instancias y más de 120 millones de *facts* acerca de esas entidades.

Yago está disponible de forma gratuita y en diversos formatos, e incluso puede ser utilizada como servicio web por otras aplicaciones que realicen consultas sobre RDFS.

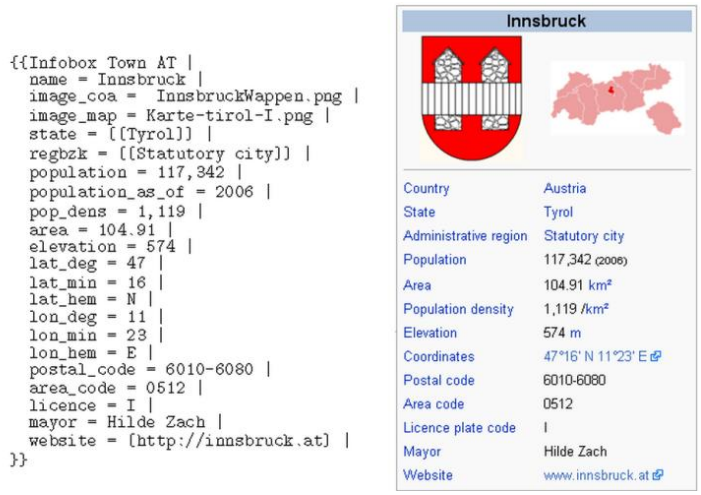
Wikipedia

Wikipedia se ha convertido en la última década en un recurso muy útil como fuente de datos para diversas aplicaciones. No sólo resulta útil la clasificación de contenido textual y multimedia en categorías semánticas, sino que éstas además forman una red semántica,

[6] Pueden consultarse en http://framenet.icsi.berkeley.edu/

utilizada por ejemplo para crear la base de datos Yago. Cuenta además con información estructurada gracias a los llamados *infobox templates* (Figura 4). Esta información puede ser directamente utilizada para el reconocimiento de entidades y relaciones, entre otras muchas aplicaciones. Concretamente, Kazama y Torisawa [2007] aplican conocimiento extraído de esta fuente para mejorar el reconocimiento de entidades.

Con el fin de extraer información estructurada de Wikipedia, y facilitar su acceso a través de la Web, surge en 2007 el proyecto DBpedia [Bizer et al., 2009a]. Este proyecto es desarrollado en colaboración entre la Universidad Libre de Berlín, la Universidad de Leipzig y OpenLink Software. Actualmente las bases de datos disponibles cuentan con más de 3.5 millones de elementos (364.000 personas, 462.000 localizaciones, etc.). La información se encuentra representada mediante RDF, que puede ser descargado bajo licencia Free Software o consultado directamente a través de servicios web mediante SPARQL. Adicionalmente, los recursos de DBpedia están enlazados con otras fuentes de datos libremente disponibles (Figura 5), creando una red de conceptos relacionados, unívocamente identificados mediante URIs. Esta red , que constituye buena parte del proyecto Linked Open Data del Consorcio W3C [Bizer et al., 2009b], favorece el acceso, la recuperación y la interoperabilidad de la información, utilizada entre otras aplicaciones para la desambiguación de entidades [Cucerzan, 2007].



```
{{Infobox Town AT |
  name = Innsbruck |
  image_coa =  InnsbruckWappen.png |
  image_map = Karte-tirol-I.png |
  state = [[Tyrol]] |
  regbzk = [[Statutory city]] |
  population = 117,342 |
  population_as_of = 2006 |
  pop_dens = 1,119 |
  area = 104.91 |
  elevation = 574 |
  lat_deg = 47 |
  lat_min = 16 |
  lat_hem = N |
  lon_deg = 11 |
  lon_min = 23 |
  lon_hem = E |
  postal_code = 6010-6080 |
  area_code = 0512 |
  licence = I |
  mayor = Hilde Zach |
  website = [http://innsbruck.at] |
}}
```


Figura 4. Ejemplo de *infobox* correspondiente a un artículo sobre la ciudad de Innsbruck. A la izquierda aparece el código fuente.

Por último, Wikipedia gracias a su contenido semi-estructurado, se ha convertido desde 2007 en el corpus utilizado para la evaluación de diversas tareas de recuperación de información dentro de la iniciativa INEX (*Initiative for the Evaluation of XML retrieval*) [Demartini et al., 2009]. También es utilizado para la creación de *gold standards* para la captura de entidades de nombre [Nothman et al., 2009] [Richman & Schone, 2008].

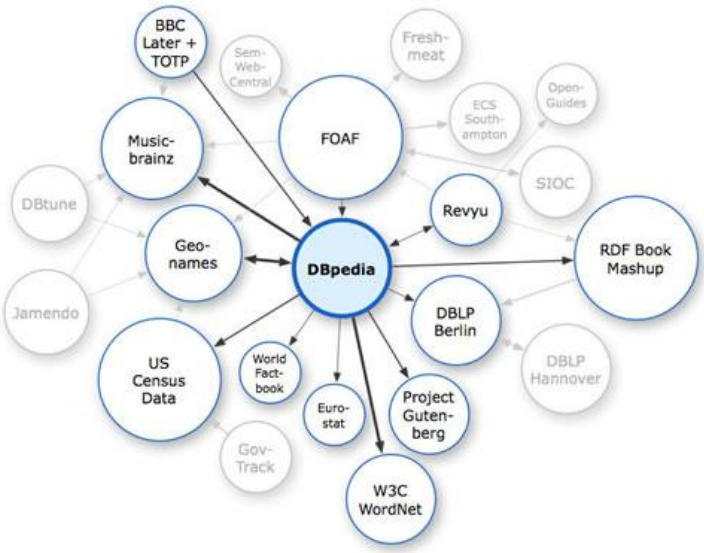


Figura 5. Relaciones entre DBpedia y otras fuentes de datos (Fuente: wiki.dbpedia.org/Interlinking).

2.4. Representación de patrones

Como hemos visto en la Sección 2.2, las técnicas para el reconocimiento de entidades construyen patrones con los que las entidades de interés encajan. Independientemente de la técnica empleada, podemos encontrar diferentes tipos de patrones para este fin ya desde los inicios de las conferencias MUC. La tarea inicial de estas conferencias fue *template-filling*, que consistía en localizar determinados elementos componentes de diferentes tipos de eventos. Por ejemplo, en un corpus sobre incidentes terroristas en América Latina, localizar el terrorista, la víctima, instrumentos y localización del ataque. Esta tarea condicionó la representación de las reglas utilizadas en los sistemas participantes a reglas *muli-slot*, es

decir, con el objetivo de identificar varios elementos de información integrantes de un escenario más complejo. Sin embargo, estos sistemas generalmente pueden ser utilizados en la identificación de un único elemento, como sistemas *single-slot*, y por tanto los modelos resultantes pueden ser aplicados en el área de Reconocimiento de Entidades.

AutoSlog [Riloff & Lehnert, 1993] es uno de estos sistemas. Utiliza reglas en forma de frames para la identificación de cada concepto de interés, que son llamados *concept nodes*. Estos nodos resultan activados en el texto por la aparición de determinadas palabras (*trigger words).* Un nodo contiene por tanto una *trigger word*, el evento al que se suscribe, el concepto a reconocer, la ubicación del mismo en la frase (e.g. sujeto, objeto directo, etc.) y una serie de restricciones relacionadas con la sintaxis (Código 3).

```
CONCEPT NODE
Name: target-subject-passive-verb-bombed
Trigger: bombed
Variable Slots: (target (*S* 1))
Constraints: (class phys-target *S*)
Constant Slots: (type bombing)
Enabling Conditions: ((passive))
```

Código 3. *Concept Node* identificado por AutoSlog para la localización de objetivos de bombas.

Un sistema similar, CRYSTAL [Soderland et al., 1995], logra reglas más generales gracias al uso de jerarquías semánticas para generalizar términos. En este caso son utilizadas jerarquías propias, pero otras herramientas han utilizado para ello WordNet. Es el caso de RAPIER [Califf & Mooney, 2004], con patrones que consisten en tres partes: el contexto izquierdo, el núcleo y el contexto derecho. Para cada una de estas partes se indican conjuntos de restricciones sintácticas (POS) y semánticas (categoría de WordNet) de la palabra o palabras que deben ocupar esa posición. En una línea similar, aunque más flexible, Whisk [Soderland, 1999] representa los patrones con ayuda de operadores de expresiones regulares, donde además pueden utilizarse sub-expresiones e información sintáctica (Código 4).

```
ID::3
Pattern:: * (Person) * ‘@Passive’ *F ‘named’ * {PP *F (Position) *
‘@succeed’ (Person)
Output:: Succession {PersonIn $1} {Post $2} {PersonOut $3}
```

Código 4. Ejemplo de patrón en Whisk para identificar eventos de reemplazo de puestos de trabajo: @Passive indica la forma de la palabra, *Person* es una sub-expresión, @succeed es un *stem* mientras *F se refiere a cualquier cadena pero de la misma categoría sintáctica.

El sistema (LP)2 [Ciravegna & Wilks, 2003a] representa el texto de forma secuencial y modela las reglas en forma de condición/acción, donde la acción consiste en determinar dónde ubicar la etiqueta de inicio o fin. Las reglas se forman indicando valores de ciertos atributos predefinidos para los tokens en una secuencia: palabra, lema, categoría léxica, mayúsculas/minúsculas y categoría semántica, esta última obtenida de listados o sistemas NER (Tabla 6).

Tabla 6. Regla obtenida por (LP)2. Los elementos en negrita forman parte de la regla obtenida a partir de la generalización de la secuencia de texto.

Word index	Word	Lemma	LexCat	Case	SemCat	Tag (action)
1	The	the	Art	Low		
2	Seminar	Seminar	Noun	Low		
3	at	**At**	Prep	Low		**stime**
4	4	4	**Digit**	Low		
5	pm	pm	Other	Low	**timeid**	
6	will	Will	Verb	Low		

La proliferación de los sistemas ad-hoc dio lugar a un marco común que permitiera aprovechar los patrones elaborados por las diferentes herramientas. Así surge *Common Pattern Specification Language* (CPSL) [Appelt & Onyshkevych, 1998]. Posteriormente ha sido adoptado con diversas variaciones en diferentes plataformas [Boguraev, 2004][Drozdzynski et al., 2004][Rinaldi et al., 2005][Cunningham et al., 2013], siendo GATE una de las más conocidas. La representación en CPSL se realiza mediante una gramática formada por los siguientes elementos:

- Información de la gramática: se puede especificar el nombre de la gramática y los tipos de entrada que recibe, es decir, los tipos de anotación que conforman el alfabeto de entrada (e.g. word, named entity annotations, etc.).
- Reglas: las reglas de la gramática, identificadas por un nombre y, opcionalmente, una prioridad. Cada regla está formada por el cuerpo principal y opcionalmente por un prefijo y un sufijo. Los componentes de la regla son pares atributo-valor separados por comas. Pueden utilizarse operadores regulares para agrupar estos componentes (e.g. OR, *, +, ?). También es posible definir un componente como una llamada a una función

externa de tipo booleano, de modo que sólo si el valor devuelto es *true*, la lectura de la entrada continúa.

- Acciones: para cada regla pueden especificarse acciones, que consisten en realizar anotaciones en el texto de entrada que servirán como alfabeto de entrada en la fase siguiente. Es posible realizar anotaciones anidadas así como utilizar expresiones condicionales del tipo *if-then* para definir la acción.
- Macros: es posible predefinir reglas mediante macros, que pueden tomar parámetros de entrada. La definición de una macro es similar a la de una regla, y en el lugar donde se invoca se sustituye por los componentes de la regla. Sus acciones se añaden a las acciones en la nueva regla.

En el Código 5 podemos ver un ejemplo de gramática en CPSL con una regla que captura una palabra con lema *douglas* seguida de una palabra en mayúscula con el lema *appelt*, y anota tanto el nombre simple como el nombre completo con la etiqueta *Name*. Estas etiquetas tendrán un atributo *type* que indica si se trata de un nombre simple (*firstName)* o completo (*wholeName)*. La fase *testGrammar* indica que esta gramática forma parte de esa fase en concreto dentro de una posible composición de varias gramáticas en cascada.

```
Phase: testGrammar
Input: Word
Rule: testRule
(({Word.lemma==douglas}):label1 {Word.lemma==appelt,
Word.uppercase==true}):label2
→
label1.Name.type=firstName, label2.Name.type=wholeName
```

Código 5. Ejemplo de gramática CPSL.

Ya en la década del 2000 proliferan los clasificadores con base probabilística, matemática, de inteligencia artificial o lógica. Los más utilizados en NER son los Modelos Ocultos de Markov (HMM), los Árboles de Decisión, los Modelos de Máxima Entropía (ME), las Máquinas de Soporte Vectorial (SVM) y los Campos Aleatorios Condicionales (CRF) [Nadeau & Sekine, 2007]. En estos casos la representación de las entidades se realiza en forma de combinaciones de atributos o funciones sobre el texto a evaluar, donde los tipos de atributos son independientes del algoritmo y seleccionados en el momento de construcción del modelo (ver Sección 2.3).

```
<class id=”Monitor”>

<pattern id=”name_first_and_price_follows” type=”pattern”
cover=”0.75”>
^$name(<tok/>{0,20} $price){1,4} </pattern>

<axiom> $price_with_tax > $price_without_tax </axiom>

<attribute id=”name” type=”name” card=”1” Eng=0,70”>
<pattern id=”model_id”>
<tok case=”UC”/> | <tok type=”ALPHA” case=”CA”/> | <tok
type=”ALPHANUM|INT”/>
<value>
<pattern cover=”0,5” p=”0,8”> (LCD (monitor|panel)?)?
<pattern src=”manuf.txt”/> <pattern ref=”model_id”/>{1,2}
</pattern>
<length><distributin min=”1” max=”7”/></length>
<pattern cover=”0,5” type=”format”> has_one_parent</pattern>
<pattern cover=”0,5” type=”format”> fits_in_parent</pattern>
<pattern cover=”0,8” type=”format”> no_crossed_inline_tags </pattern>
<pattern cover=”1,0” type=”format”> no_crossed_block_tags </pattern>
</value>
<context>
<pattern p=”0,3” cover=”0,2”> model? Name:? $</pattern></context>
</attribute>
```

Código 6. Fragmento de código de una ontología de extracción para monitores en Ex.

En cualquier caso para colecciones semi-estructuradas como la Web se siguen utilizando wrappers. Los wrappers han sido definidos como "procedimientos simples de extracción de información para recursos semi-estructurados" [Kushmerick, 1997], y son utilizados por ejemplo en el sistema BWI [Freitag & Kushmerick, 2000] y en el sistema Thresher [Hogue & Karger, 2005]. El primero trabaja con secuencias de tokens o tipos de tokens (alfabético, alfanumérico, numérico, puntuación, etc.) mientras que el segundo, más orientado a la Web, trabaja con representaciones en árbol donde los nodos son etiquetas HTML. También orientado a la Web surge un modelado de patrones basado en ontologías llamado Ex [Labský et al., 2009]. Este modelado en XML representa cuáles son las clases y sus atributos mediante patrones variados: caracteres, expresiones regulares, etiquetas, referencias a otros patrones, estructura (e.g. *has_one_parent*), etc. Es posible indicar además la efectividad de cada patrón en términos de precisión/exhaustividad y definir axiomas que establecen restricciones acerca del valor de los atributos. En el Código 6 podemos ver un ejemplo con algunas de las características principales: indica que en el 75% de los casos una instancia de la clase monitor comienza con un atributo *name* de hasta 20 tokens, seguido de hasta cuatro

atributos de tipo *price*. Especifica el patrón de *name* y la precision (p) y recall (*cover*). Un axioma establece que el atributo *price_with_tax* ha de ser mayor que *price_without_tax.*

En los últimos años se han desarrollado lenguajes destinados especialmente a hacer escalable la creación de escenarios (extracción *multi-slot*) a partir de datos individuales (*single-slot)*. Es el caso por ejemplo de xlog [Doan et al., 2006], basado en predicados, y AQL [Chiticariu et al., 2010b], basado en álgebra relacional. Pero en estos casos las entidades que componen los escenarios continúan siendo capturadas mediante listados o expresiones regulares. Estas últimas continúan siendo la aproximación usada en algunos trabajos de NER [Brauer et al., 2011] [Wu & Pottenger, 2005].

2.5. Tipos de herramientas

Encontramos al menos dos grandes grupos de herramientas relacionadas con el reconocimiento de entidades: aquellas que tienen como objetivo el reconocimiento automático de entidades, y aquellas que sirven de soporte para la anotación de entidades en un documento. Estas últimas son llamadas herramientas de anotación semántica. La anotación semántica no es exclusiva del área de reconocimiento de entidades, pero es frecuente en este ámbito el reconocimiento de entidades con cierto grado de automatización, por lo que las técnicas que se utilizan son similares. En cualquier caso, es más frecuente el uso de técnicas semi-supervisadas en la anotación semántica, interviniendo el usuario activamente en el etiquetado de los textos.

2.5.1. Herramientas de reconocimiento automático de entidades

Muchas de las herramientas que encontramos habitualmente para realizar la tarea de reconocimiento automático de entidades utilizan aprendizaje supervisado. Generalmente encontramos esta técnica unida al uso de recursos como listados, heurísticas y expresiones regulares, que determinan si una palabra pertenece a un listado, cumple determinada regla

o encaja con un patrón predefinido. Algunos ejemplos de este tipo de herramientas son Freeling (Atserias, Casas et al. 2006), Afner (Molla, Van Zaanen et al. 2006), Supersense Tagger (Ciaramita, Altun 2006) y TextPro (Pianta, Girardi et al. 2008). La herramienta Annie (*A Nearly New IE system*), incorporada en GATE [Cunningham et al., 2013], utiliza autómatas en cascada (dispone de un lenguaje basado en CPSL llamado JAPE) y listados, es capaz de resolver co-referencia y además añadir plug-ins para aumentar su potencia (MinorThird para aprendizaje supervisado, wrappers, etiquetadores gramaticales, *stemmers*, soporte para ontologías, etc.).

En cuanto a las herramientas comerciales, destacan las empresas BBN Technologies, IBM (herramienta EROCS), Trifeed, Inxight (herramienta ThingFinder[7]) y Temis. También Alias-I y ClearForest Ltd. disponen de herramientas NER (LingPipe y Calais respectivamente), que además se ofrecen gratuitamente bajo ciertas condiciones para fines de investigación e incluso pueden utilizarse como servicios web.

Otras dos herramientas relacionadas con el reconocimiento de entidades son Google Squared y Wolfram Alpha[8]. La primera, desaparecida en 2011 (aunque su tecnología es posiblemente usada para mejorar las búsquedas), era capaz de generar series de conceptos similares a los introducidos (similar a la herramienta *KnowItAll* en uno de sus modos de funcionamiento). Por ejemplo, si el usuario escribe "yellow" y "red" la herramienta devuelve otros conceptos como "green", "black", etc. y los atributos de cada uno de ellos. También es posible mostrar conceptos pertenecientes a una clase. Por ejemplo, al introducir la palabra "colors", la herramienta devolvería los conceptos anteriores. Además el usuario puede añadir nuevas instancias e incluso nuevos atributos que la herramienta va aprendiendo.

Wolfram Alpha no incorpora técnicas automatizadas para la captura de entidades, sino que estas han sido almacenadas previamente (hasta el momento más de 10 billones de datos factuales sobre más de 1000 dominios) sobre el que funciona un motor de

[7]Permite capturar diversos tipos de entidades y relaciones en multitud de idiomas. También es posible generar manualmente patrones de captura, determinando los valores de una variedad de atributos de las entidades (atributos lingüísticos, tipográficos, de pertenencia a listados, etc.).
[8] http://www.wolframalpha.com

recuperación capaz de procesar consultas en lenguaje natural y realizar complejas operaciones sobre la información recuperada. Algunos de los datos que almacena son referentes a entidades concretas (por ejemplo, una empresa, lugar geo-político o persona concreta).

Existen también herramientas específicas de dominio, especialmente en biomedicina. Algunas de las entidades más frecuentemente reconocidas son proteínas, genes, ADN, ARN, línea celular, tipo celular, mutaciones y propiedades de las estructuras proteicas. Las técnicas utilizadas para su reconocimiento son similares a las de dominio general, salvo que en este caso la dependencia de corpus anotados, listados y ontologías es mayor debido a la ausencia de patrones fijos fácilmente reconocibles en este tipo de entidades. Esto convierte este campo en uno de los más ricos en recursos anotados semánticamente, aunque con los mismos problemas de estandarización que en otros. Algunas de las herramientas específicas para el reconocimiento de entidades biomédicas son Abner (proteínas, ADN, ARN, línea y tipo celular), AbGene (genes y proteínas), PIE y BIORAT (proteínas y sus interacciones) (Marrero, et al., 2010)[Marrero et al., 2012a] .

2.5.2. Herramientas de anotación semántica

La anotación semántica formalmente identifica conceptos y relaciones entre conceptos en documentos, y está dirigida principalmente a su uso por máquinas. Es imprescindible para la Web Semántica, pero también para la gestión de conocimiento en otros contextos. Las herramientas de anotación semántica deben cumplir ciertos requisitos necesarios a la hora de desarrollarlas [Uren et al., 2006]:

- Formatos estándar: el etiquetado de recursos es costoso, por lo que la estandarización permitiría su posterior aprovechamiento. La adecuación a estándares puede permitir además la independencia de formatos propietarios y la colaboración entre organismos. Entre los estándares útiles para esto encontramos OWL y RDF.
- Usabilidad: interfaces usables que faciliten la tarea y además permitan la anotación colaborativa y control de acceso.

- Soporte de ontologías: deben dar soporte a múltiples ontologías, asegurar consistencia entre anotaciones y referencias ante posibles variaciones en la ontología, y permitir la navegación y edición sobre las mismas.
- Soporte de formatos heterogéneos de documentos: los estándares de la Web Semántica suelen asumir que los documentos a anotar están en formatos web como HTML o XML. Este es el caso por ejemplo de los frameworks Annotea [Kahan et al., 2002] y CREAM [Handschuh & Staab, 2002]. Sin embargo, la anotación de documentos en múltiples formatos es necesaria para que sea útil su incorporación en cualquier proceso de trabajo.
- Cambios sobre los documentos: control de anotaciones realizadas en documentos que están siendo o van a ser modificados.
- Almacenamiento de las anotaciones: en general, en el entorno de la Web Semántica se prefiere desacoplar el documento de su anotación, pero en otros entornos más centrados en el documento se prefiere mantenerlos juntos, quizá con diferentes vistas.
- Automatización: etiquetado automático para disminuir los costes asociados al etiquetado de grandes colecciones. Para ello es necesario incorporar tecnologías de extracción de información.

Es precisamente en este último punto donde son útiles en este ámbito las técnicas de reconocimiento de entidades, que generalmente son de tipo semi-supervisado, partiendo de ejemplos ya etiquetados o patrones ya conocidos por el usuario. Algunos ejemplos de herramientas de anotación que incorporan estas técnicas son Thresher [Hogue & Karger, 2005] y Melita [Ciravegna & Wilks, 2003b].

Thresher es capaz de etiquetar instancias y atributos de una clase mediante la generación, a partir de ejemplos, de *wrappers* aplicados a la Web; estos *wrappers* son además almacenados en formato RDF para facilitar su reutilización. Melita por su parte alterna fases de entrenamiento, test y producción, creando nuevas anotaciones a partir del etiquetado del usuario de forma progresiva. Se entrena a partir de documentos ya etiquetados por el usuario y las anotaciones obtenidas del último modelo obtenido son mostradas al usuario para que las corrija y así mejorarlo. Para el entrenamiento utiliza el sistema (LP)2, utilizado en el reconocimiento automático de entidades. También en el

dominio biomédico es posible encontrar herramientas de anotación específicas para el mismo. Algunas de estas herramientas son MetaMap [Aronson, 2001], Mgrep [Xuan et al., 2007], OBA [Jonquet et al., 2008] y CONANN [Reeve & Han, 2007].

2.6. Principales foros de evaluación

A nivel internacional, las principales evaluaciones sobre reconocimiento de entidades tuvieron lugar hasta 2008 en las Conferencias *Message Understanding Conferences* (MUC), *Conference on Natural Learning* (CoNLL) y *Automatic Content Extraction* (ACE). Actualmente las conferencias con tareas en activo relacionadas con NER son *INitiative for the Evaluation of XML Retrieval* (INEX), *Text Analysis Conference* (TAC) y *Text REtrieval Conference* (TREC).

2.6.1. Message Understanding Conferences

El término *Named Entity* es usado por primera vez en la sexta Conferencia MUC [Grishman & Sundheim, 1996], una de la serie de conferencias financiadas por la agencia DARPA para el desarrollo de técnicas para la extracción de información. Los participantes en estas conferencias debían rellenar plantillas con información de textos procedentes de noticias o informes de interés militar (e.g. terrorismo, accidentes aéreos o de misiles, etc.), indicando por ejemplo la causa, agente, fecha y lugar de un evento. Concretamente, para la tarea de reconocimiento de entidades se usaron textos del *Wall Street Journal* (actualmente disponibles a través del *Linguistic Data Consortium*[9] -LDC-). Los tipos de entidades a reconocer eran personas, localizaciones, organismos, fechas y cantidades numéricas. Los resultados eran medidos en términos de precisión y exhaustividad sobre dos objetivos diferentes: la identificación de entidades y la clasificación. La primera únicamente es considerada válida si los límites de la entidad son correctos, mientras que la clasificación se considera válida cuando el tipo asignado sea el correcto aunque los límites no sean exactos

[9] http://www.ldc.upenn.edu/

(mientas se solapen). El resultado final es la F ($\beta = 1$) calculada sobre todos los tipos de entidad y sobre los dos objetivos (micro-Averaged[10]), donde la F (también llamada F-measure) es definida como:

$$F = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

mientras precision y recall se definen como:

$$precision = \frac{relevantes\ recuperados}{recuperados} \qquad recall = \frac{relevantes\ recuperados}{total\ relevantes}$$

Se obtuvieron tasas muy elevadas de precision y recall, generalmente superiores al 90%. Concretamente el mejor sistema obtuvo un 96% de recall y un 97% de precision.

Tabla 7. Corpus utilizado en CoNLL 2003. El corpus en inglés procede de noticias periodísticas de *Reuters*, y el corpus en alemán procede de noticias periodísticas del periódico *Frankfurter Rundshau*.

		Artículo	Frases	Tokens	LOC	MISC	ORG	PER
Inglés	**Corpus**	946	14987	203621	7140	3438	6321	6600
	Corpus desarrollo	216	3466	51362	1837	922	1341	1842
	Corpus test	231	3684	46435	1668	702	1661	1617
Alemán	**Corpus**	553	12705	206931	4363	2288	2427	2773
	Corpus desarrollo	201	3068	51444	1181	1010	1241	1401
	Corpus test	155	3160	51943	1035	670	773	1195

2.6.2. Conference on Natural Learning

Las conferencias CoNLL se desarrollaron desde 1999 hasta 2008 de la mano del grupo en aprendizaje del lenguaje natural de la *Association for Computational Linguistics* (ACL). La temática varía cada año, y durante los años 2002 y 2003 estuvo dedicada al reconocimiento de entidades en diversos lenguajes (español y holandés en 2002, y alemán e inglés en 2003) [Sang, 2002] [Sang & Meulder, 2003]. Los tipos de entidades a reconocer son persona, organismo, localización y miscelánea. La evaluación se realiza en base a la F (β =1), de nuevo micro-averaged, considerando válido un resultado únicamente si coincide en el tipo y en los límites. Los corpus de entrenamiento y test proceden de artículos periodísticos y

[10] Una medida micro-averaged se calcula a partir de la mezcla de los resultados de todas las clases, en este caso, los tipos de entidad. Una medida macro-averaged por el contrario se calcula como la media aritmética de los resultados obtenidos para cada tipo de entidad.

son de libre acceso (Tabla 7). En el caso del corpus en inglés, es proporcionado por el NIST previa petición, aunque está sujeto a limitaciones en su uso.

Participaron un total de 16 sistemas, con una media cercana al 90% el mejor sistema, y de algo más del 60% el peor (Tabla 8). Todas las herramientas utilizaron técnicas de aprendizaje automático supervisado con corpus anotados.

Tabla 8. Resultados sobre el corpus de test de las herramientas que compitieron en CoNLL 2003.

Inglés				Alemán			
Sistema	**Precision**	**Recall**	**F**	**Sistema**	**Precision**	**Recall**	**F**
Florian	88.99%	88.54%	88.76±0.7	Florian	83.87%	63.71%	72.41±1.3
Chieu	88.12%	88.51%	88.31±0.7	Klein	80.38%	65.04%	71.90±1.2
Kelin]	85.93%	86.21%	86.07±0.8	Zhang	82.00%	63.03%	71.27±1.5
Zhang	86.13%	84.88%	85.50±0.9	Mayfield	75.97%	64.82%	69.96±1.4
Carreras (a)	84.05%	85.96%	85.00±0.8	Carreras (a)	75.47%	63.82%	69.15±1.3
Curran	84.29%	85.50%	84.89±0.9	Bender	74.82%	63.82%	68.88±1.3
Mayfield	84.45%	84.90%	84.67±1.0	Curran	75.61%	62.46%	68.41±1.4
Carreras (b)	85.81%	82.84%	84.30±0.9	McCallum	75.97%	61.72%	68.11±1.4
McCallum	84.52%	83.55%	84.04±0.9	Munro	69.37%	66.21%	67.75±1.4
Bender	84.68%	83.18%	83.92±1.0	Carreras (b)	77.83%	58.02%	66.48±1.5
Munro	80.87%	84.21%	82.50±1.0	Wu	75.20%	59.35%	66.34±1.3
Wu	82.02%	81.39%	81.70±0.9	Chieu	76.83%	57.34%	65.67±1.4
Whitelaw	81.60%	78.05%	79.78±1.0	Hendrickx	71.15%	56.55%	63.02±1.4
Hendrickx	76.33%	80.17%	78.20±1.0	De Meulder	63.93%	51.86%	57.27±1.6
De Meulder	75.84%	78.13%	76.97±1.2	Whitelaw	71.05%	44.11%	54.43±1.4
Hammerton	69.09%	53.26%	60.15±1.3	Hammerton	63.49%	38.25%	47.74±1.5
Baseline	71.91%	50.90%	59.61±1.2	Baseline	31.86%	28.89%	30.30±1.3

2.6.3. Automatic Content Extraction

El objetivo del programa ACE [Doddington et al., 2004], patrocinado por el NIST (*U.S. National Institute of Standards and Technology*), es el desarrollo de tecnología para el procesamiento automático de texto. Las tareas se centran en la identificación de entidades, relaciones y eventos. El tipo de entidades consideradas son persona, organización, localización, instalación (*facility*), arma, vehículo y entidad geo-política (en ediciones previas al 2008 se consideraban también las entidades de tipo valor y expresiones temporales como tareas adicionales). La evaluación es muy compleja, puesto que no sólo se ponderan los diferentes tipos de entidades, sino que también se valora la identificación del

subtipo de entidad (la ACE predefine una lista de subtipos para cada tipo de entidad), la clase (genérica, específica, negativa e insuficientemente especificada) y la identificación del tipo de mención (entidad o co-referencia nominal o pronominal). Las identificaciones parciales son tenidas en cuenta únicamente si la parte principal de la entidad es reconocida al menos en una cierta proporción de caracteres. Aunque basada en medidas de precisión y exhaustividad, la penalización ante diversos tipos de error es variable, lo que resulta en un método poco intuitivo y problemático a la hora de establecer comparativas [Nadeau & Sekine, 2007].

Tabla 9. Resultados de los sistemas en la competición ACE'08 para la tarea de Reconocimiento Local de Entidades (*Local Entity Detection and Recognition*) sobre corpus en inglés (los porcentajes negativos son debidos al sistema de penalización utilizado).

Sistema	**Total**	***Broadc. Convers.***	***Broadc. News***	***Meetings***	***News***	***Telep.***	***Usenet***	***Weblogs***
IBM	50.8%	44.6%	37.7%	-11.9%	58.1%	26.1%	25.5%	51.0%
BBN Technologies	52.6%	42.0%	36.9%	-44.2%	61.3%	22.1%	31.1%	54.8%
Fudan University	-17.6%	-45.1%	-43.3%	-441.2%	9.0%	-197.0%	-48.4%	4.4%
Pontificia Univ. Catolica do Rio de Janeiro, Genesis Inst.	-46.3%	-54.7%	-21.1%	-57.6%	-64.9%	-18.6%	-48.8%	10.1%
Fondazione Bruno Kessler	-90.0%	-148.1%	-98.8%	-404.6%	-63.8%	-436.2%	-83.0%	-43.5%
AU-KBC Research Center	-269.1%	-340.0%	-279.7%	-911.4%	-188.3%	-999.9%	-279.6%	-177.4%

En cuanto a los corpus utilizados, éstos pertenecen a diversos géneros (noticias periodísticas, blogs, foros, registros de conversaciones telefónicas, etc.) y se encuentran en cuatro idiomas: inglés, chino, árabe y español. Todos ellos están disponibles previo pago a través de la LDC. En la Tabla 9 se muestran las puntuaciones finales sobre los diferentes corpus en inglés para los participantes en la tarea de reconocimiento de entidades en la competición de 2008 [National Institute of Standards and Technology (NIST), 2008].

2.6.4. Otros foros relacionados con reconocimiento de entidades

A partir de 2008 comienzan a llevarse a cabo otros foros con tareas relacionadas con el reconocimiento de entidades, aunque difieren de las tradicionales.

INEX Entity Ranking

En 2008 y 2009 se lleva a cabo una nueva tarea relacionada con la identificación y clasificación de entidades que utiliza como corpus Wikipedia. Esta tarea es denominada *INEX Entity Ranking Track* (XER) [Demartini et al., 2009] y se lleva a cabo en el marco de INEX (*INitiative for the Evaluation of XML Retrieval*). La tarea consiste en la ordenación por relevancia de entidades que respondan a una pregunta, tarea conocida como "ranking de entidades". En este caso la entrada o preguntas al sistema consisten en describir aquello que queremos recuperar e información adicional, como la categoría o un listado de entidades de ejemplo (Tabla 10). La salida consiste en la ordenación de los artículos de Wikipedia contenidos en la colección INEX Wikipedia que traten de las entidades buscadas ordenados por relevancia. Como medida de evaluación se utiliza MAP (*Mean Average Precision)*. Esta medida calcula la media de *Average Precision* (AP) obtenida por cada sistema por cada pregunta. AP puede definirse como:

$$AP = \frac{\sum_{k=1}^{n}(P(k) \cdot rel(k))}{número\ de\ documentos\ relevantes\ en\ la\ colección}$$

donde k es la posición en la secuencia de documentos recuperados ante una pregunta, $P(k)$ es la precision en el corte k, y $rel(k)$ es un función que devuelve 1 si el documento en la posición k es relevante, y 0 en caso contrario. MAP entonces puede definirse para cada sistema sobre el conjunto de preguntas de tamaño Q como:

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}$$

Con esta medida, muy utilizada en las evaluaciones en TREC, se obtiene en esta tarea unas tasas máximas de 0.341 y 0.517 en 2008 y 2009 respectivamente.

Tabla 10. Ejemplo de pregunta de entrada en INEX Entity-Ranking Track.

Identificador	"9999"
Título	Impressionist art in the Netherlands
Descripción	I want a list of art galleries and museums in the Netherlands that have impressionist art
Narrativa	Each answer should be the article about a specific art gallery or museum that contain impressionist or post-impressionist art works.
Categoría	art museums and galleries
Entidad	Van Gogh Museum
Entidad	Kröller-Müller Museum

Text Analysis Conference

Las Conferencias TAC (*Text Analysis Conference*) comienzan en 2008 bajo la organización del NIST. En 2009 surge la tarea KBP (*Knowledge Base Population*) [National Institute of Standards and Technology (NIST), 2009], con el objetivo de completar información específica sobre entidades de diversos tipos (e.g. alias, causa de muerte o fecha de nacimiento para entidades de tipo persona). El fin de esta tarea es poblar ontologías a partir de información no estructurada.

La tarea principal es la llamada *Slot Filling*, donde el objetivo es localizar los valores de los atributos de una entidad dada. En este caso, a partir de un nombre de entidad, un texto donde se cite, su tipo, el identificador del nodo si existe en la base de datos y una lista de atributos que opcionalmente pueden ser ignorados, se debe dar como resultado el valor de los atributos obligatorios y los documentos de donde se obtuvieron (Código 7).

Las respuestas son evaluadas como correctas, incorrectas (si la respuesta o el documento de donde procede no es válido) o inexactas (cuando es incompleta o incluye texto no apropiado) por personal de la LDC y desde el 2010 se utilizan las medidas precision, recall y F-measure. En cuanto a los corpus, se utiliza Wikipedia (concretamente las infoboxes) como corpus de referencia, y el corpus sobre el que trabajan, recopilado por la LDC, está formado por más de un millón de artículos periodísticos.

```
<query id="SF1">
<name>John Doe</name>
<docid>SUN-009</docid>
<enttype>PER</enttype>
<nodeid>E101</nodeid>
<ignore>per:date_of_birth per:place_of_birth per:religion</ignore>
</query>
```

Código 7. Ejemplo de topic para la tarea *Slot Filling* en TAC.

TREC Entity Track

En 2009 surge un nuevo *track* en TREC llamado *Entity Track*, con el objetivo de promover el desarrollo de buscadores capaces de devolver las páginas principales de las entidades buscadas. Por ejemplo, ante la pregunta "Aerolíneas que actualmente usan aviones Boeing 747", las páginas resultantes deberían ser las páginas web principales de las aerolíneas que

cumplan con este requisito [Balog et al., 2010]. Las preguntas se representan mediante un nombre de entidad, su página principal (o una de ellas), el tipo de entidad que se pretende recuperar y una descripción de la relación de la entidad proporcionada con la que se pretende recuperar (Código 8).

Hasta ahora el tipo de entidades objetivo soportadas son únicamente persona, organismo, producto y localización. En cuanto a la colección de trabajo, se trata de un subconjunto del corpus ClueWeb09, que contiene aproximadamente 50 millones de páginas en inglés.

La evaluación es realizada siguiendo la metodología habitual en TREC: se toma un pool de 10 salidas de cada sistema, que son evaluadas por el NIST. Las páginas son evaluadas como no relevantes, relevantes y primarias. La medida principal de evaluación es NDCG@R, donde R es el número de páginas primarias y relevantes.

```
<query>
<num>7</num>
<entity_name>Boeing 747</entity_name>
<entity_URL>clueweb09-en0005-75-2292</entity_URL>
<target_entity>organization</target_entity>
<narrative>Airlines that currently use Boeing 747 planes.</narrative>
</query>
```

Código 8. Ejemplo de topic en el TREC Entity Track.

En 2010, además de la tarea descrita llamada *Related Entity Finding* (REF), se añade la tarea *Entity List Completion* (ELC) como tarea piloto. Esta tarea traslada las búsquedas de la Web a la Web Semántica utilizando como colección Linked Open Data [Bizer et al., 2009b].

Capítulo 3
Evaluación de herramientas NER

Only through observation will you perceive weakness –Charles Darwin

3.1. Introducción

Para obtener una visión más completa del área de reconocimiento de entidades se han evaluado diversas herramientas NER, tanto desde el punto de vista de sus características como de su efectividad y errores más frecuentes.

Los sistemas de reconocimiento de entidades nombradas son evaluados a través de competiciones como MUC, CoNLL y ACE, y más recientemente XER y ET. Sin embargo las causas de error de las técnicas empleadas han sido poco estudiadas. Como consecuencia, se ha realizado un estudio para determinar cuáles son las características de las entidades que más influyen en los errores cometidos por herramientas NER de propósito general y así establecer posibles líneas de investigación para corregir ciertos patrones de error comunes. Adicionalmente se pretenden obtener datos acerca de las características y tasas de efectividad de las herramientas disponibles dado que:

- Como vimos en Sección 2.6, las métricas de evaluación son diferentes en cada competición: varían de la simplicidad de CoNLL a la complejidad de ACE. Por ejemplo,

en CoNLL se exige identificación exacta en la clasificación mientras que MUC es más flexible y son computadas también las identificaciones parciales para calcular las tasas de clasificación. Por otro lado, la evaluación en ACE se basa en un complejo algoritmo difícil de interpretar y de comparar con otras competiciones [Nadeau & Sekine, 2007]. Además, las primeras competiciones, con tasas de F-measure en torno 0.9 la primera y 0.8 la segunda, sugieren un área resuelta [Cunningham, 2005], mientras que las tasas de la ACE, en torno al 50% [National Institute of Standards and Technology (NIST), 2008] sugieren lo contrario.

- La evaluación de los sistemas NER se limita a aquellos que se presentan a las competiciones. Pero estos sistemas no están necesariamente disponibles ni a nivel de investigación ni comercialmente. Con los actuales recursos, la comparativa con herramientas que no acuden al evento resulta en muchos casos imposible porque no han sido entrenadas para el reconocimiento de los tipos de entidades que se evalúan. Y aún siendo así, el concepto que subyace detrás de cada entidad varía entre las distintas competiciones (Marrero et al., 2009)[Marrero et al., 2009b].
- Los grandes corpus etiquetados pueden favorecer a las herramientas que disponen de listados más amplios y sin embargo no implica necesariamente una calidad superior de la herramienta. Aunque las evaluaciones en CoNLL sí ofrecen información acerca de las evaluaciones con ayuda de listados o sin ella, no ocurre lo mismo con los restantes foros [Marrero et al., 2009a][Marrero et al., 2009b].

Para realizar este estudio se ha usado un documento de prueba con entidades típicas (personas, organismos, localizaciones, fechas y cantidades) y otras de tipos marginales reconocidos por algunas herramientas. Se evalúa la capacidad de identificación (es o no entidad) y clasificación (qué tipo de entidad) de modo independiente en términos de precisión y exhaustividad, y se analizan asimismo los errores más frecuentemente cometidos.

3.2. Herramientas NER evaluadas

Son muchas las herramientas que se han localizado a través de referencias en trabajos científicos o documentación comercial. Para este estudio se han establecido los siguientes criterios para seleccionar las herramientas NER a evaluar:

- Debe permitir el procesamiento de textos externos a la aplicación.
- Debe funcionar de modo independiente y no como un entorno de desarrollo/framework. Es decir, no debe requerir que el usuario aporte recursos propios, tales como listados u ontologías, para su funcionamiento.
- Debe procesar textos en un idioma común, ya que éste condiciona las técnicas aplicadas. El idioma seleccionado ha sido el inglés por ser ampliamente utilizado y existir mayor número de herramientas.
- Ha de ser de propósito general (no adscritas en exclusiva a un dominio específico), tanto comercial como de investigación, y que ofrezca al menos demos completas que cumplan los requisitos anteriores.

Algunas herramientas han sido descartadas por no cumplir los requisitos exigidos. Es el caso de Trifeed[11], que sólo admite artículos periodísticos predeterminados, y otras muy populares del ámbito de la biomedicina como AbGene[12], Abner[13] y BioNer[14]. En otros casos se han eliminado herramientas debido a su escasa eficacia o falta de mantenimiento. Por último, dos herramientas con buenos resultados en las competiciones no se han considerado por no disponer de una versión gratuita o de prueba (EROCS de IBM y la herramienta NER de BBM Technologies).

Finalmente se evaluarán las siguientes herramientas: LingPipe, ClearForest, Annie, Afner, Supersense (con los tres modelos de entrenamiento que incluye: Modelo CoNLL, Modelo WNSS, Modelo WSJ), TextPro y YooName:

[11] http://www.trifeed.com/
[12] ftp://ftp.ncbi.nlm.nih.gov/pub/tanabe/AbGene/
[13] http://pages.cs.wisc.edu/~bsettles/abner/
[14] http://isoft.postech.ac.kr/Research/BioNER/POSBIOTM/NER/main.html

- LingPipe[15] es un conjunto de librerías en Java desarrolladas por la empresa Alias-i para el procesamiento del lenguaje natural. Inicialmente está preparada para la detección y clasificación de entidades de persona, organismo y localización para el inglés, pero también es posible entrenarla mediante corpus para otros idiomas. Además de la detección y clasificación de entidades, ofrece otras funcionalidades como la corrección ortográfica o la clasificación de textos en inglés. Dispone de una interfaz y diversas demos con las que es posible probar textos. Se trata de una herramienta de uso gratuito para fines de investigación, aunque es posible su compra con fines comerciales.
- ClearForest SWS[16] es una herramienta comercial de la empresa ClearForest Ltd., actualmente adquirida por Reuters. Permite analizar textos en inglés e identificar los tipos ENAMEX, además de algunos otros como productos, monedas, etc. Se ha construido un servicio web sobre parte de la herramienta para la captura de entidades llamado Gnosis. Se trata de un plug-in gratuito para el navegador Mozilla Firefox que captura numerosos tipos diferentes de entidades. También ofrecen una API para uso Web que puede utilizarse gratuitamente bajo ciertas condiciones. Actualmente ha evolucionado hacia una herramienta llamada Calais, que entre otros muchos servicios adicionales, permite establecer relaciones entre las entidades y detectar eventos y roles. Está disponible en Web, pero no permite adjuntar ficheros para su análisis. Por este motivo la evaluación solo se ha realizado con la demo web de ClearForest SWS.
- Annie [Cunningham et al., 2013] es un módulo para extracción de entidades incorporado en el framework GATE de código abierto y bajo licencia GNU desarrollado en la Universidad de Sheffield. Está implementado en Java, e incorpora como plug-ins y librerías recursos tanto propios como ajenos para diversos aspectos relacionados con el procesamiento del lenguaje natural (e.g. Lucene, MinorThird, Google, Weka etc.). Puede utilizarse como API, pero también provee su propia interfaz para su uso independiente. Annie se ofrece como un módulo con un conjunto de recursos por defecto (separación en tokens, analizador sintáctico, listados, etc.) que pueden ser utilizados en combinación para la captura de entidades o sustituidos por otros plug-ins. La evaluación de la

[15] http://www.alias-i.com/lingpipe/
[16] http://sws.clearforest.com/

herramienta se ha realizado a partir de los recursos que incorpora por defecto en el procesamiento de textos en inglés.

- Afner [Molla et al., 2006] es una herramienta NER de código libre bajo licencia GNU, desarrollada en C++ en la Universidad de Macquaire. Actualmente se utiliza como parte de otra herramienta de *Question Answering* llamada AnswerFinder, por lo que prioriza maximizar la exhaustividad. Afner puede utilizarse también como API de otras aplicaciones o de modo independiente. Utiliza un modelo de aprendizaje supervisado que incluye, entre otras características, la pertenencia a un listado o el uso de expresiones regulares. Es posible añadir listados y expresiones regulares, y entrenar nuevos modelos. Por defecto, es capaz de reconocer nombres de personas, organismos, localizaciones, miscelánea, cantidades monetarias y fechas en textos en inglés. Dispone de varios modelos ya generados, aunque utiliza uno por defecto.
- Supersense Tagger [Ciaramita & Altun, 2006] es una herramienta de código abierto desarrollada en C++ con licencia Apache License v2.0. Está diseñada para desambiguación e identificación de entidades mediante el uso de las categorías de WordNet en sustantivos y verbos. Reconoce personas, organismos, localizaciones, expresiones temporales y de cantidad. Utiliza aprendizaje automático supervisado, y dispone de tres modelos ya entrenados: CoNLL, WSJ y WordNet.
- TextPro Tools Suite [Pianta et al., 2008] está desarrollada en C++ en el *Centro per la ricerca scientifica e tecnologica* (ITC-irst), en Trento. Ofrece diversas funcionalidades de NLP conectadas a modo de pipeline. Utiliza aprendizaje automático y gazetteers. Funciona para el inglés y el italiano, y dispone de una demo web para ambos idiomas. Está sujeto a licencia GNU.
- YooName [Nadeau et al., 2006] es una herramienta desarrollada en la Universidad de Ottawa. Incorpora técnicas de aprendizaje semi-supervisado aplicadas a la Web que permiten identificar entidades a partir de una clasificación predefinida de nueve tipos de entidades: persona, organismo, localización, miscelanea, instalación (*facility*), producto, evento, elemento natural y unidad. Estos tipos se subdividen a su vez en alrededor de 100 subtipos. En el momento de realización de este trabajo existía una versión web con

una demo[17] donde se podían escribir textos en inglés para ser analizados. Actualmente la herramienta forma parte de la empresa InfoGlutton[18].

Como se puede observar en la Tabla 11, la mayoría de las herramientas seleccionadas han sido desarrolladas en C++, y ofrecen interfaz de consola y API. Con respecto al grado de conocimiento informático necesario para su uso, la mayoría de ellas han sido clasificadas como avanzadas, y sólo una como simple (simple indica que es suficiente descargar y ejecutar el fichero, mientras que avanzada se refiere a que es necesario llevar a cabo procesos más complejos, como incluir librerías adicionales, compilar, configurar, etc.). En cuanto al número de tipos de entidades reconocidas, este varía de 3 a más de 100 (incluyendo subtipos) y la técnica más habitual para el reconocimiento es el uso de modelos generados a partir de clasificadores.

Tabla 11. Características de las herramientas. El símbolo de interrogación indica que no ha sido posible encontrar información al respecto.

Herramienta	Lenguaje Program.	Interfaz	Licencia	Dificultad de uso	Demo	Tipos de entidad reconoc.	Técnica
Supersense-CoNLL	C++	Consola/API	Apache 2.0	Avanzada	No	4	Modelo clasificación
Supersense-WordNet	C++	Consola/API	Apache 2.0	Avanzada	No	27	Modelo clasificación
Supersense-WSJ	C++	Consola/API	Apache 2.0	Avanzada	No	> 100	Modelo clasificación
Afner	C++	Consola/API	GNU	Avanzada	No	6	Modelo clasificación
Annie	Java	Gráfica/API	GNU	Avanzada	Sí	~12	Listados y gramáticas predefinidas
TextPro	C++	Consola/API	GNU	Avanzada	Sí	4	Modelo clasificación
YooName	?	?	?	?	Sí	>100	Listados generados por bootstrapping
ClearForest	?	Web/API	Commerc.	?	Sí	6	?
Lingpipe	Java	API	Free/Develop./Startup	Simple	Sí	3	Modelo clasificación

[17] http://yooname.wordpress.com/2008/03/
[18] http://www.infoglutton.com/yooname-named-entity-recognition.html

3.3. Metodología

Para analizar la efectividad de las herramientas así como los errores más frecuentes en el reconocimiento de entidades se ha elaborado un documento de prueba para su evaluación. Se han identificado algunos de los factores que pueden resultar más influyentes en NER por figurar habitualmente como atributos en las herramientas basadas en aprendizaje automático. Entre ellos tenemos las características tipográficas, propiedades léxico-semánticas y utilización de recursos lingüísticos. En concreto los factores analizados son (Tabla 12):

- Uso de mayúsculas: mayúscula en la primera letra y alternancia mayúscula/minúscula en el texto de la entidad. Además se diferencia si son o no inicial de frase.
- Uso de comillas.
- Palabras compuestas (con y sin uso de guiones en ellas).
- Pertenencia a listados (estimado por tratarse de términos muy habituales cuya inclusión en listados es frecuente).
- Reconocible por patrones regulares de caracteres (e.g. fechas y cantidades monetarias).
- Palabras polisémicas.

Tabla 12. Características analizadas y número de entidades en el documento con cada característica.

Características	**Número de entidades**
Inicial de frase	31
Primera mayúscula	48
Todo minúscula	13
Todo mayúscula	15
Mayúsculas alternativas	10
Comillas	14
Compuestas	39
Conocidas o reconocibles con expresiones regulares	56
Polisémicas	11
Inventadas (imposible captura por listados o expresiones regulares)	5

Estos factores han sido cruzados de forma equilibrada con los diferentes tipos de entidad analizados, en los que predominan los tipos comunes a todas las herramientas (Tabla 13). Por ejemplo se construyeron frases (siempre lingüísticamente correctas) con entidades que suelen ser reconocidas y clasificadas mediante listados (e.g. Spain), incluyendo entidades

con elementos tipográficos especiales como el guión (e.g. se incluye "Papua-New Guinea" y "Papua New Guinea"). También entidades conocidas en inglés son comparadas con otras completamente inventadas, donde la ayuda de listados no pueda haber sido determinante. Como resultado, el documento contiene un total de 579 palabras, distribuidas en 13 párrafos y 53 frases, en las que se han concentrado 100 ocurrencias de entidades de varios tipos y características.

Tabla 13. Tipos de entidades distribuidas en el documento de evaluación.

Tipo de entidad	**Número de ocurrencias**
Localización	44
Persona	32
Organismo	12
Moneda	4
Fecha	2
Porcentaje	2
Numero	2
Otro	2

Los análisis han sido realizados teniendo en cuenta los tipos que cada herramienta potencialmente puede reconocer. Dada la ambigüedad en torno a las semánticas que cada tipo incluye (dos etiquetas iguales pueden no significar lo mismo), los subtipos en los que se divide y el modo de etiquetar diferente de cada herramienta, se ha optado por analizar manualmente los resultados. De este modo en lugar de adaptar la herramienta a los corpus previamente etiquetados, como ocurre en los foros de evaluación, en este estudio se ha adaptado el documento y la evaluación de resultados a las herramientas objeto de estudio. El análisis de los resultados se ha realizado siguiendo un doble enfoque:

- Comparativa de la eficacia de las herramientas en identificación y clasificación de entidades, analizando cada proceso de modo independiente. Se han utilizado las medidas precision y recall, y micro-average F, teniendo en cuenta tanto identificaciones completas como parciales. Para ello se han considerado únicamente los tipos de entidad que cada herramienta puede potencialmente reconocer.
- Comparativa de los errores más frecuentes y las características tipográficas, léxicas o semánticas que presentan. Este análisis ha sido realizado mediante el estudio de los errores cometidos por cada herramienta y las características presentes en los mismos.

3.4. Eficacia de las herramientas

Los resultados de las medidas precision y recall tanto en identificación completa como en clasificación (Figura 6) están en general por encima de 0.5. Las excepciones son la herramienta Afner, tanto en precision como en recall, y los valores de recall de la herramienta YooName. ClearForest destaca en su comportamiento por obtener tasas de precision superiores a 0.9, aunque otras herramientas como Supersense Tagger y Annie logran valores que, aunque menos elevados, son superiores a 0.7 y resultan más equilibrados con respecto a las tasas de recall.

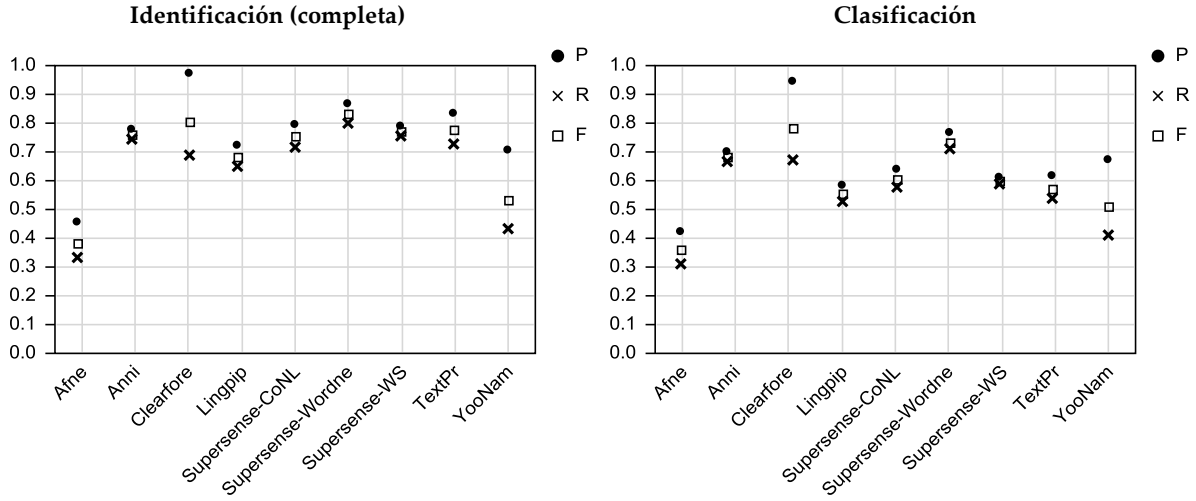


Figura 6. Precision, recall y F en identificación completa y clasificación de entidades.

Dado que la clasificación es un proceso dependiente de la identificación de entidades, la F de identificación es siempre superior a la de clasificación (Figura 7). Las diferencias más notables entre ambos procesos se aprecian en las herramientas TextPro y en menor medida en SupersenseTagger – Modelo WSJ, que destacan en los procesos de identificación pero no en la clasificación de las entidades que previamente han logrado identificar. Como contrapartida, Afner, ClearForest y YooName consiguen clasificar adecuadamente prácticamente todo aquello que identifican.

A diferencia de los resultados obtenidos en MUC y CoNLL, con tasas generalmente superiores a 0.9 en el primero, y superiores a 0.8 en el segundo, en este caso se observa una

F de clasificación muy variable, de entre el 0.3 y 0.8, que tan sólo en dos casos supera el 0.7 y en 6 de los 9 casos está por debajo o rozando valores de 0.6.

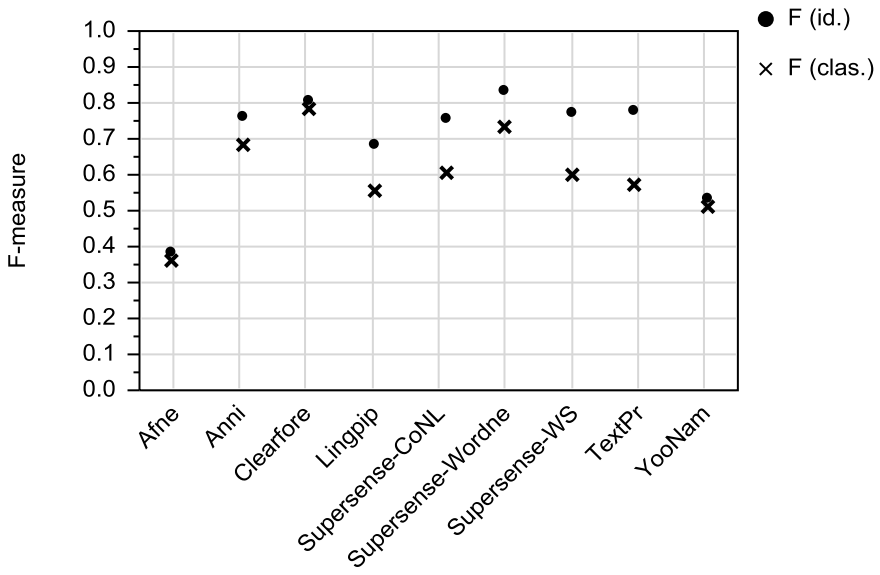


Figura 7. Relación entre la F obtenida en identificación completa y en clasificación.

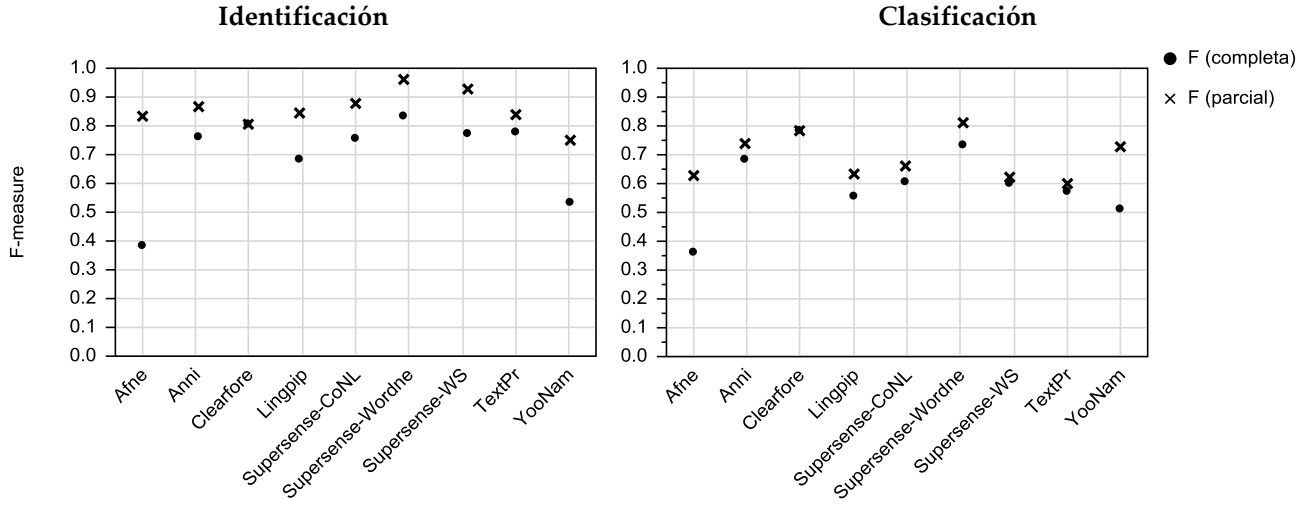


Figura 8. Relación de la F en la identificación y clasificación considerando identificación completa o parcial.

Cuando en lugar de identificaciones completas consideramos válidas las identificaciones parciales, los resultados en identificación mejoran una media de 0.16 (Figura 8). Herramientas como ClearForest o TextPro prácticamente no mejoran, lo que indica su capacidad para delimitar acertadamente las entidades. El caso opuesto lo encontramos en Afner, con una mejora absoluta de 0.45 en identificación parcial. Trasladando estos datos a

la clasificación, las diferencias son menos notables y sólo destacan las herramientas YooName y Afner, con mejorías absolutas de 0.21 y 0.27 respectivamente al no exigir delimitación exacta de las entidades.

3.4.1. Resultados por tipo de entidad

Tabla 14. Resultados por tipo de entidad.

Herram.	Entidad	Nº	F	Herram.	Entidad	Nº	F
Supers. CONLL	Person	32	0.63	YooName	Person	33	0.30
	Location	43	0.64		Location	44	0.51
	Org.	13	0.72		Org.	13	0.88
	Miscelanea	4	0		Vocation	4	1
Supersense-WordNet	Person	38	0.65		Country	21	0.66
	Location	48	0.78		State/Prov.	4	0.75
	Group	25	0.88		City	7	0.70
	Time	9	0.66		Loc (other)	11	0
	Quantity	9	0		Company	12	0.08
	Food	6	1		Month	2	1
	Communic..	1	1		Week Day	2	1
	Cognition	2	0.66		Food	6	1
	Substance	1	0		Mineral	1	0
	Relation	1	0		Vegetal	1	1
	Plant	6	1	Clear Forest	Person	53	0.72
	Object	1	1		Country	19	0.97
	Other	1	1		State/Prov.	6	1
Supersense-WSJ	Person	32	0.30		City	10	0.18
	Person-Desc.	5	1		Company	12	0.95
	Geo-Pol.(other)	20	0.10	Text Pro	Person	32	0.59
	Country	18	0.70		Location	44	0.51
	State/Province	6	0.66		Org.	13	0.88
	Geo-Pol-Desc.	9	1	Afner	Person	32	0.50
	Corporation	12	0.69		Location	44	0.36
	Org.-Descrip.	12	0.86		Org.	12	0
	Date	10	1		Date	2	0.50
	Money	4	0.28	Annie	Person	32	0.73
	Food	7	1		Person-title	2	1
	Ordinal	1	1		Location	44	0.62
	Cardinal	7	0.92		Organizat.	12	0.81
Lingpipe	Person	32	0.67		Date	5	1
	Location	47	0.47		Money	4	0.33
	Org.	12	0.78		Percent	2	1

En la Tabla 14 vemos los tipos reconocidos por las herramientas y sus tasas de efectividad sobre el documento de evaluación. Se puede observar la importante diferencia entre el número de tipos de entidad reconocidos sobre un mismo texto, con herramientas que reconocen tan sólo 3 tipos de entidad y otras capaces de reconocer 14. Además, las tasas de efectividad sobre cada tipo pueden variar notablemente respecto al comportamiento global. Es el caso de Afner, que globalmente tiene peores resultados que el resto pero está por encima de Supersense-WSJ o de YooName en el reconocimiento de personas.

También observamos grandes diferencias entre las tasas obtenidas para unos tipos y para otros en una misma herramienta. Por ejemplo, la herramienta ClearForest presenta una F de 0.18 para el caso de ciudades y de 0.95 para el caso de nombres de compañías, teniendo en ambos casos un número de ocurrencias similar (10 y 12 concretamente).

3.5. Errores más frecuentes

Para determinar cuáles son los errores más frecuentemente cometidos por las herramientas se ha realizado un análisis sobre todas las entidades reconocidas y las características de cada una de ellas. A estos datos se han añadido aquellos segmentos de texto que al menos una de las herramientas ha reconocido como entidad, además de todos los adverbios de tiempo y sustantivos encontrados en el documento. De este modo se ha creado una colección con todos los elementos que potencialmente pueden haber sido reconocidos como entidad. El resultado ha sido la anotación de 137 elementos diferentes por cada herramienta, con el valor correspondiente para cada uno de los atributos estudiados y los resultados de identificación completa, identificación parcial y clasificación (calculada sólo sobre las entidades identificadas correctamente, ya sea parcial o completamente). También aquí el análisis ha sido personalizado por cada herramienta, puesto que se ha considerado que la no identificación de un elemento por una herramienta es un error sólo cuando esa herramienta concreta puede reconocer el tipo de entidad al que pertenece (en caso de duda se ha optado por considerar que no existe error).

Los atributos estudiados son los ya descritos en la Sección 3.3:

- Inicial frase: el elemento se encuentra al inicio de una frase (por tanto la primera letra está en mayúscula).
- Inicial mayúscula: la primera letra está en mayúscula.
- Mayúscula: todo el elemento está en mayúscula.
- Mayúsculas alternativas: se alternan minúsculas y mayúsculas (generalmente son palabras compuestas, una en minúscula y otra en mayúscula).
- Comillas: el elemento o parte de él está entre comillas.
- Compuesta: elemento compuesto por varios tokens.
- Polisémica: palabra polisémica.
- Recursos: entidad susceptible de ser reconocida gracias al uso de recursos como listados (por ser muy conocida o porque su número es limitado, como los días de la semana) o expresiones regulares (e.g. cantidades monetarias, porcentajes, etc.).

Los resultados pueden observarse en la Tabla 15 de forma global, y en la Tabla 16 por herramienta. Se ha aplicado el test de Fisher para determinar si existen diferencias estadísticamente significativas entre las proporciones de errores cuando las entidades presentan distintos valores del atributo. En cada casilla de las tablas se muestra el porcentaje de errores cuando el atributo no se cumple y cuando sí, en este orden, además del nivel de significatividad. En el caso del atributo "Recursos", dado que sus valores no se limitan a positivo o negativo, sino que existen elementos para los que no podemos afirmar si son fácilmente reconocibles mediante el uso de recursos externos, se aplica un test Chi-Square con dos grados de libertad, en lugar de Fisher. Los resultados en las tablas en este caso mostrarán por orden el porcentaje de errores cuando el elemento no puede encontrarse en recursos (porque es inventado), no se sabe, y cuando sí es muy probable que se encuentre en listados u otros recursos o sea reconocible por expresiones regulares.

Los resultados de la Tabla 15 muestran que sí existe una importante dependencia de la tipografía por parte de las herramientas analizadas. Aquellos elementos con la inicial en mayúscula tienden a ser peor identificados, especialmente de forma completa. Esto se debe a que en general los elementos escritos en minúscula no son reconocidos como entidad

acertadamente, pero si una palabra común está en mayúscula, las herramientas tienden a identificarla erróneamente como entidad. También la escritura de elementos completamente en mayúscula parece desconcertar a las herramientas, y cometen más errores en este caso.

Tabla 15. Influencia de los atributos en el porcentaje de errores en identificaciones parciales, identificaciones completas y clasificaciones. Los números indican el porcentaje de errores cuando el atributo no se cumple y cuando sí. Diferencias significativas al * 0.05, ** 0.01 y *** 0.001.

	I.Frase	**I.May.**	**May.**	**May.Alt.**	**Comillas**	**Comp.**	**Polis.**	**Recurs.**
Id. Parcial		11-15 **	11-28 ***				11-36 ***	21-9-14 ***
Id. Completa		13-29 ***	19-39 ***	19-45 ***	20-30 **	16-34 ***	19-36 ***	44-9-23 ***
Clasificación		9-24 ***	19-29 *			16-26 **	18-44 ***	32-10-15 ***

En cuanto a la alternancia de mayúsculas y minúsculas (generalmente palabras compuestas) se observa que sólo afecta a la identificación completa. Esto es porque las herramientas suelen etiquetar como entidad únicamente aquella parte del elemento identificable mediante recursos o con la inicial en mayúscula. Con las comillas ocurre exactamente lo mismo: suele etiquetarse aquella parte entrecomillada de las palabras compuestas, dejando fuera el resto. Estos factores influyen además en el atributo de palabra compuesta, que muestra ser significativo también para identificación completa.

El atributo relacionado con el uso de recursos resulta también influir en identificación y clasificación. Como se esperaba, se observa que la proporción de identificaciones y clasificaciones correctas es mejor cuando la entidad es conocida o reconocible por listados o expresiones regulares que cuando se trata de una entidad inventada. También la polisemia afecta tanto a la identificación como a la clasificación, aunque negativamente.

En cuanto a los resultados por tipo de herramienta, vemos que Supersense-Wordnet, Supersense-WSJ, Annie y TextPro son las herramientas afectadas por menos atributos de las entidades. En el primer caso sólo se observan errores significativos sobre entidades en minúscula mientras que Supersense-WSJ muestra problemas con las palabras polisémicas. En el caso de Annie y TextPro se observa cierta dependencia de los listados o expresiones regulares, especialmente en Annie, donde el porcentaje de errores cometidos ante palabras

reconocibles por recursos como listados o expresiones regulares es notablemente menor respecto de aquellas que no lo son.

Tabla 16. Influencia de los atributos en el porcentaje de identificaciones incorrectas (parcial o completa) por herramienta. Los números indican el porcentaje de identificaciones incorrectas cuando el atributo no se cumple y cuando sí. Diferencias significativas al * 0.05, ** 0.01 y *** 0.001.

	I.Frase	**I.May.**	**May.**	**May.Alt.**	**Comillas**	**Comp.**	**Polis.**	**Recurs.**
Afner			9-48 ***				81-31 *	26-4-19 ***
Annie								88-4-30 ***
Clearforest	12-34 **	8-24 **					12-56 ***	36-9-14 ***
Lingpipe			10-38 **	12-36 *				
Supers-WSJ							9-31 *	
Supers-CoNLL			8-33 **				7-50 ***	
Supers-WNet		11-3 *						
TextPro								26-9-10 *
YooName	18-35 *	9-33 ***	19-37 *				17-69 ***	38-9-29 ***

En siguiente lugar encontramos las herramientas LingPipe, Afner y Supersense-CoNLL, todas ellas con problemas para el reconocimiento de palabras escritas completamente en mayúsculas, muy especialmente Afner. Por otro lado, LingPipe tiene problemas además con las palabras que alternan mayúscula y minúscula, Supersense-CoNLL y Afner muestran importantes problemas para el reconocimiento de palabras polisémicas, mientras que Afner tiene mejores resultados con palabras susceptibles de pertenecer a listados o reconocibles mediante expresiones regulares.

Por último, ClearForest y YooName tienden a etiquetar como entidad aquellas palabras que empiezan por mayúscula, y muestran problemas también con aquellas que se encuentran en mayúscula por ser inicio de frase. YooName presenta además problemas con palabras escritas completamente en mayúscula (una observación más directa de sus resultados nos permite observar que en general la variación en el uso de mayúsculas de una misma entidad a lo largo del texto, impide que se anote como tal). En ambos casos la polisemia afecta negativamente mientras el uso de recursos lo hace positivamente.

3.6. Discusión

Las tasas de F obtenidas en clasificación se encuentran entre 0.3 y 0.8, con 6 de las 9 herramientas por debajo o alrededor de 0.6. Estas tasas muestran una importante diferencia respecto a las tasas observadas en MUC y CoNLL, donde el peor sistema en ambos foros logra un 0.6 de efectividad, mientras la amplia mayoría de sistemas superan el 0.8 (concretamente 40 de los 46 sistemas evaluados en ambos foros). Aún teniendo en cuenta identificaciones parciales, más de la mitad de las herramientas se sitúan entre 0.6 y 0.7. Por tipos de entidad, los resultados muestran casos en los que las tasas obtenidas para tipos específicos no guardan relación con la posición global de esa herramienta entre las restantes, así como herramientas con importantes diferencias entre los tipos reconocidos.

Si aislamos los procesos de identificación y clasificación, vemos que existen casos donde se aprecia un excepcional buen comportamiento del algoritmo de clasificación (e.g. Afner) o lo contrario (e.g. TextPro, Supersense-WSJ). Además, no todas las herramientas funcionan igual en lo que a delimitación de la entidad se refiere. En el caso de Afner y YooName los resultados empeoran notablemente al exigir identificación completa. Esto es un aspecto a considerar si la tarea requiere una delimitación exacta de las entidades. ClearForest, por el contrario, delimita adecuadamente prácticamente todo aquello que identifica.

En cuanto al análisis de errores, se observa que las herramientas en general confían en el uso de mayúscula para la identificación de las entidades, salvo si es inicio de frase (excepto ClearForest y YooName). También tienen menos dificultades para reconocer adecuadamente entidades potencialmente reconocibles por listados o uso de expresiones regulares, aunque las herramientas que aplican únicamente modelos de aprendizaje automático (LingPipe y Supersense) no se ven afectadas por esta característica. Además se observan problemas en la identificación exacta de entidades compuestas, que puede venir dada por la confianza en listados (localización en un listado de una parte de la entidad) unida a las variaciones tipográficas (alternancia de mayúscula/minúscula, comillas) en este tipo de entidades. Por último, la polisemia también ha resultado ser un problema en la identificación y clasificación de entidades.

Las tasas de efectividad de las herramientas no parecen tener una relación directa con la influencia de un mayor o menor número de atributos en el reconocimiento. Por ejemplo, ClearForest y Supersense-Wordnet obtienen las mejores tasas de clasificación y sin embargo ClearForest muestra errores por aspectos tipográficos y semánticos (uso de mayúscula, polisemia, recursos) mientras que los errores en Supersense-Wordnet tan sólo parecen indicar problemas en el reconocimiento de entidades en minúscula (probablemente porque esta última, a diferencia de ClearForest, es capaz de reconocer más tipos de entidad que habitualmente se escriben en minúscula). En cualquier caso, la combinación de los datos referentes a los errores, tipos de entidad reconocidos y sus características, y tasas de efectividad obtenidas permiten hacernos una idea del comportamiento de una herramienta ante otros corpus y entidades específicas.

3.7. Conclusiones

Se ha realizado una evaluación de diversas herramientas NER con el triple objetivo de analizar las características de las herramientas, los errores que cometen ante ciertas características de las entidades y conocer las tasas de efectividad ante un texto del dominio general.

Hay que tener en cuenta que esta evaluación está sesgada por las características del documento usado en la evaluación (en realidad, como ocurre en cualquier evaluación), que en este caso presenta deliberadamente un tamaño reducido para hacer viable el estudio comparativo de diversas herramientas con tipos de entidad y modos de etiquetado heterogéneos. Su pequeño tamaño puede reducir la validez de los resultados, aspecto que se ha intentado paliar con la distribución homogénea de los tipos y características de las entidades que aparecen en él, y con estudios de significatividad adecuados en el caso del análisis de errores.

Los resultados muestran importantes variaciones entre las herramientas. Sólo partiendo de sus características, resulta llamativo que herramientas NER de dominio general como las

consideradas, aplicadas a un mismo texto, de tamaño reducido y dominio general, puedan reconocer de 3 a 14 tipos de entidad diferentes. Esto hace difícil saber qué esperar de una herramienta NER de este tipo y abre la puerta a un análisis más detallado sobre el significado de entidad y la tarea de reconocimiento de entidades.

En cuanto a su efectividad, los resultados muestran tasas alejadas en muchos casos del 0.9 de F logrado en MUC y CoNLL que han llevado incluso a considerar la tarea resuelta [Cunningham, 2005]. Por otro lado, la metodología seguida para la evaluación de la efectividad ha contemplado por separado los procesos de identificación y clasificación (sólo MUC lo hizo), y además se ha tenido en cuenta la identificación parcial. De este modo se ha podido observar que no todas las herramientas se comportan igual ante cada uno de estos procesos. También existen grandes diferencias respecto a la efectividad lograda para cada uno de los tipos de entidad, incluso en una misma herramienta. Estos aspectos han de tenerse en cuenta a la hora de seleccionar una herramienta en función de las necesidades de la tarea. Lo mismo ocurre con los errores estudiados a partir de las características de las entidades. Aunque hemos comprobado que el uso de mayúscula es generalmente condicionante en la identificación de entidades, no todas las herramientas presentan esta cualidad, sino que depende en gran medida de las características de los tipos de entidad que reconoce y las tecnologías que usa. Esto hace poco adaptables las herramientas a nuevos tipos de entidad y dominios, salvo mediante la incorporación de nuevos modelos de aprendizaje en aquellas que lo soporten, lo que supone anotar nuevos corpus.

Hasta donde llega nuestro conocimiento, no existen en la literatura estudios de este tipo, por lo que además de los objetivos propuestos, el trabajo aquí realizado es útil para establecer una comparativa entre algunas de las herramientas NER actualmente disponibles, y además podría servir de base para elaborar una metodología de selección de herramientas NER ante proyectos concretos, tratando de suplir algunas de las carencias que presentan los foros internacionales en este sentido.

Capítulo 4
Falacias y retos en Reconocimiento de Entidades Nombradas

Fallacies do not cease to be fallacies because they become fashions –G. K. Chesterton

4.1. Introducción

Como vimos en el capítulo anterior, la variedad de tipos de entidad reconocidos por herramientas NER de propósito general hace difícil saber qué esperar de una herramienta de este tipo, y abre la puerta al análisis del concepto de entidad. Esta cuestión no ha sido analizada aún, aunque ha sido manifestada en otras palabras: el concepto de *Named Entity* (NE) apareció en un contexto de aplicaciones de NLP y está lejos de estar lingüísticamente claro y asentado [Borrega et al., 2007].

Por otro lado, las tasas de efectividad de las herramientas analizadas están muy por debajo de las obtenidas en los foros MUC y CoNLL. Esto contrasta con el convencimiento general de que NER es una tarea resuelta, con tasas de éxito por encima del 95% [Cunningham, 2005]. En este capítulo se demuestra que esto es una falacia, y se muestra

cómo la falta de acuerdo sobre el concepto de NE tiene importantes implicaciones en las herramientas NER y más especialmente en el modo en el que se evalúa la tarea. Los actuales foros relacionados con NER no resuelven este problema porque la definición de la tarea es muy distinta. La evolución de las técnicas en el área requiere reconsiderar si NER está resuelta o no, y diseñar procedimientos de evaluación adecuados a las necesidades actuales.

4.2. Evolución del significado de NE

En 1991, antes de que surgiera el término *Named Entity*, Lisa F. Rau propone un método para reconocer nombres de compañías en textos, lo que hoy es comúnmente aceptado como una tarea NER. Se dio cuenta de que estas entidades eran problemáticas para el procesamiento del lenguaje natural por ser palabras desconocidas, pero eran piezas de información relevantes para el análisis textual [Rau, 1991].

Posteriormente en 1997 se acuña el término *Named Entity* en la sexta conferencia MUC, y se define como *unique identifiers of entities* [Chinchor & Robinson, 1997]. Años después, en 2002, encontramos definiciones que lo limitan a nombres propios. Concretamente se define como *a proper noun, serving as a name for something or someone* [Petasis et al., 2000]. Los autores justifican esta restricción simplemente porque existe un número significativo de nombres propios en los corpus. Por otro lado, Alfonseca y Manandhar definen la tarea de NER como *the task of classifying unknown objects in known hierarchies that are of interest for us for being useful to solve a particular problem* [Alfonseca & Manandhar, 2002]. Esta aproximación ha sido seguida en la jerarquía BBN [Brunstein, 2002], en la ontología GENIA para recuperación de información biomédica, y en tareas de QA [Kim et al., 2003].

Como vimos en la Sección 2.1.2, otra definición de NE es la propuesta en 2007 por Nadeau y Sekine, que se apoya en el concepto de designador rígido: *named aims to restrict the task to only those entities for which one or many rigid designators, as defined by Kripke, stand for the referent* [Nadeau & Sekine, 2007][Nadeau, 2007].

A pesar de la variedad de definiciones de NE, ha habido cierto acuerdo sobre las semánticas a reconocer en los principales foros de evaluación. Entre 1996 y 2008, la evaluación de las herramientas NER se ha llevado a cabo a nivel internacional en los foros MUC, CoNLL (*Conference on Computational Natural Language Learning*) y ACE (*Automatic Content Extraction*). La definición de la tarea en todos ellos era bastante similar: dada una semántica, el objetivo era localizar elementos en el texto que respondieran a ella. Personas, organismos y localizaciones eran tipos de NE siempre presentes en estos foros, mientras que expresiones temporales y numéricas eran específicas de MUC y ACE, y el tipo *miscelánea* era específico de CoNLL. ACE incorporó además armas, instalaciones, vehículos y entidades geo-políticas, además de subdivisiones en numerosos subtipos. La evaluación se realizaba en todos ellos mediante corpus anotados, generalmente de pequeño tamaño (comparados con los grandes corpus utilizados en recuperación de información) y propios del dominio periodístico en su mayor parte.

Actualmente hay otras conferencias relacionadas con NER, aunque tratan tareas que difieren de las tradicionales: INEX Entity Ranking track (XER) [Demartini et al., 2009], TREC Entity Track (ET) [Balog et al., 2010] y TAC Knowledge Base Population task (KBP) [National Institute of Standards and Technology (NIST), 2009]. XER arrancó en 2007 con el objetivo de ordenar por relevancia artículos de Wikipedia de las entidades de interés ante ciertas preguntas y/o entidades similares. ET empezó en 2009 para promover el desarrollo de motores de recuperación capaces de recuperar las páginas principales de la entidad buscada, actualmente restringidas a personas, organismos, localizaciones y productos. KBP comienza como una tarea en las conferencias TAC con el objetivo de completar atributos relacionados con una entidad dada, que puede ser una persona (e.g. alias, cause of death and date of birth), un organismo o una localización geo-politica. En este último caso la tarea está más cerca de la identificación de relaciones que del reconocimiento de entidades.

En cuanto a las herramientas NER, se han realizado diversos proyectos en diferentes ámbitos donde han surgido nuevas categorías de NE's. Es el caso de los nombres de genes y proteínas, por ejemplo, de gran importancia en la recuperación de información en el dominio biomédico (Marrero et al., 2010)[Marrero et al., 2012a]. Adicionalmente, las

herramientas NER comerciales ofrecen cada vez más tipos de NE predefinidos, además de la posibilidad de adaptarse a nuevos tipos exigidos por los usuarios. Por ejemplo, la familia de productos Calais, de Thomson Reuters, actualmente reconoce 39 tipos diferentes de entidades, entre los que podemos encontrar programas de televisión y de ligas deportivas. Thing Finder, otra importante herramienta comercial de la empresa Inxight, reconoce 45 tipos predefinidos, entre los que encontramos algunos tan poco frecuentes como los índices financieros.

4.3. ¿Qué es una Entidad Nombrada?

Hemos visto que los expertos en Reconocimiento de Entidades Nombradas han dado diferentes definiciones sobre el significado de Entidad de Nombre. Un análisis de estas definiciones nos permite categorizarlas según los siguientes criterios: categoría gramatical, designación rígida, identificación única y dominio de aplicación. Para determinar cuál o cuáles de estos factores caracterizan a una entidad de nombre hemos seleccionado y analizado varias de ellas de diversas fuentes (Tabla 17).

Tabla 17. Ejemplo de NEs extraídas de la jerarquía de Sekine (SH), y de los corpus anotados o las guías de MUC, CoNLL03, ACE y GENIA (GEN). Los símbolos de interrogación indican dependencia del contexto.

Entidad	Nombre propio	Design. Rígido	Id. único	Ejemplo de dominio/objetivo	Fuente
water	No	Yes	? (con gas o natural)	Química	SH: natur. obj-mineral
$10million	No	?	? (dólares americanos o mexicanos)	Finanzas	MUC-7: Money
whale	No	?	? (orca u otra)	Biología	SH: sea animal
Bedouins	No	?	? (personas específicas o grupo)	Defensa/Noticias	CoNLL03: Miscellanea
Airbus A310	Sí	?	? (avión específico)	Defensa/Tecnol.	CoNLL03: Miscellanea
Batman	Sí	Sí	? (gente disfrazada)	Películas/Disfraces.	ACE: Person
Cytokine	Sí	?	? (proteína específica)	Biomedicina	GEN:protein family
twelve o'clock noon	No	?	?(día concreto)	Noticias	MUC-7: Time

4.3.1. Categoría gramatical: nombre propio

Las entidades de nombre han sido definidas como nombres propios, o nombres comunes actuando como nombres propios, porque su categoría gramatical designa realidades únicas con función identificativa. Sin embargo, observamos que algunos términos de la Tabla 17 no son nombres propios, como es el caso de *water* y *whale.* Algunas de las características de los nombres propios, como la ausencia de inflexiones o determinantes, la falta de significado léxico y el uso de mayúsculas, son insuficientes para describir entidades nombradas. Esto ya ha sido apuntado en los trabajos para el etiquetado del corpus ANCORA: *the classic grammatical approach to proper noun analysis is surely insufficient to deal with the problems NERC poses* [Borrega et al., 2007]. Por tanto, esta característica es fácilmente descartable como criterio para la definición de NEs.

4.3.2. Designador Rígido

Algunos autores continúan citando la definición de designación rígida dada por Kripke como una parte esencial de la definición de NE, aunque reconocen que esta definición es flexibilizada en ocasiones por razones prácticas [Nadeau & Sekine, 2007]. La capacidad de identificación de los designadores rígidos es explicada por Kripke con ejemplos como *Richard Nixon*, que es un identificador rígido a diferencia de *President of the United States of America*, que no lo es porque el referente cambia cada ciertos años. También *Hesperus* y *Phosphorus* son designadores rígidos, puesto que ambos tienen como referente al planeta Venus en todos los posibles mundos donde Venus existe, y no designan ninguna otra cosa en otro mundo. Estos conceptos provienen de Filosofía del Lenguaje y tienden a ser bastante controvertidos [LaPorte, 2011]. Por ejemplo, existe un acuerdo general sobre aceptar *Phosphorus* y *Richard Nixon* como designadores rígidos, aunque actualmente *Phosphorus* también designa el elemento químico (que en realidad fue nombrado porque resplandecía como el planeta Venus), y *Richard Nixon* puede referirse a muchas personas que comparten el mismo nombre, no solamente al expresidente. Siguiendo a Kripke, incluso términos que no son habitualmente considerados como entidades de nombre, como

gold, *hot*, *redness* o *loudness*, son designadores rígidos, así que la caracterísitica de designación rígida no es lo suficientemente clara para caracterizar a las NE. De hecho, excepto para casos que han sido explícitamente designados como tales, no podemos asegurar que los ejemplos de la Tabla 17 cumplan esta restricción.

4.3.3. Identificación única

Relacionado con la designación rígida, otro modo de definir NEs según la literatura científica es con el concepto de identificación única. Las Conferencias MUC establecen que los elementos a reconocer son identificadores únicos de entidades. Sin embargo, ya hemos visto que un mismo término habitualmente tiene diferentes referentes (e.g. *Phosporus*) y en muchas ocasiones el ente referido depende del contexto o incluso del conocimiento previo del receptor (e.g. *Taj Mahal* es también un casino de Las Vegas). Incluso los nombres propios, considerados habitualmente como NEs, no se refieren a elementos unívocos. Lo que ocurre es que su referente simplemente es más fácil de ubicar: e.g. Springfield es más fácil de localizar en un discurso sin contexto que "el centro del campo", pero sigue existiendo ambigüedad en ambos casos. Siguiendo este argumento, cualquier cosa podría ser referida unívocamente, aunque en la práctica el referente pueda resultar ambiguo (e.g. el concepto de agua no es ambiguo, pero sí su uso: podría usarse para referirse al elemento químico natural, pero también podría usarse para referirse al agua con gas o a una botella de agua).

Otro modo de considerar la identificación única es por el concepto de identificación que se utiliza en los lenguajes de programación o en las ontologías al hablar de los objetos o instancias de una clase. El lenguaje OWL (Web Ontology Language), definido por el World Wide Web Consortium (W3C), establece que los individuos en una ontología vienen definidos por la clase a la que pertenecen, como instancias de ella, mientras que la clase describe conceptos de un dominio o agrupa individuos que comparten propiedades. Sin embargo, la decisión de si un concepto particular puede ser considerado clase o instancia

depende de las aplicaciones de la ontología [Noy & McGuiness, 2001] y lo mismo ocurre en los lenguajes de programación con los objetos y sus instancias.

4.3.4. Propósito y dominio de aplicación

El propósito y dominio de aplicación han determinado desde el principio las entidades de nombre a reconocer. En las Conferencias MUC, patrocinadas por la agencia DARPA, el reconocimiento de ciertos tipos de información y no otros tenía un claro sesgo militar. Ejemplos típicos en esta serie de conferencias era determinar el agente, tiempo, causa y localización de un evento en un texto. Un claro ejemplo de este sesgo podemos verlo también en las conferencias ACE, donde son incluidas las categorías de vehículos y armas de un total de tan sólo siete categorías a reconocer.

En este sentido también el área de *Question Anwering* ha influenciado notablemente la definición de entidad de nombre. La jerarquía propuesta por Sekine, con más de 200 categorías (Figura 1), justifica su extensión porque 7 u 8 clases no son suficientes para cubrir las necesidades de las aplicaciones, refiriéndose a QA y a Extracción de Información en general, más específicamente en el ámbito periodístico [Sekine & Nobata, 2004]. Se ha manifestado incluso que es posible considerar que las NEs habitualmente responden a las 5 típicas cuestiones en el ámbito periodístico: qué, quién, cuándo, dónde y por qué. Esta característica ha sido explotada para reconocimiento de entidades gracias a la observación de cierta correlación entre ser una entidad y aparecer en determinadas posiciones en un texto [Shinyama & Sekine, 2004].

Por tanto, la definición de NE según el propósito y dominio de aplicación parece ser la única consistente tanto en la literatura como en los foros de evaluación y las herramientas, aunque obviamente estos objetivos y dominios puedan ser infinitos [Marrero et al., 2012b].

4.4. Implicaciones en la evaluación del Reconocimiento de Entidades

Los criterios de anotación de las entidades a reconocer difieren en los diferentes foros tradicionales de evaluación: MUC, CoNLL y ACE. Los aspectos más conflictivos en la anotación son los tipos de entidades a reconocer, los criterios para su identificación y anotación y su acotación en el texto (Tabla 18).

4.4.1. Tipos de entidades a identificar

Los tipos de entidades consideradas en los foros de evaluación hasta 2008 coinciden bastante, aunque son muy limitadas (Tabla 19). La variedad de tipos de entidades predefinidos actuales así como la necesidad mostrada por una adaptación a nuevas entidades sugeridas por el usuario pone de manifiesto la limitación en cuanto a semánticas ofrecida por los foros tradicionales para evaluar el área.

Tabla 18. Ejemplos obtenidos de los corpus o guías de anotación (con referencias a la norma exacta seguida) de MUC, CoNLL03 y ACE. Se muestra el diferente modo de anotación de una misma entidad y las diferencias de criterio respecto a la identificación como NE (I), sus límites (B) y la categoría asignada (C).

	MUC-7	**CoNLL03**	**ACE**	**Discrep.**
Baltimore defeated the Yankees	<Baltimore>LOC <Yankees>ORG (ref. A.1.6)	<Baltimore>ORG <Yankees>ORG	<Baltimore>NAM.ORG.SPO <Yankees>NAM.ORG.SPO (ref. 6.2)	C
Zywiec Full Light	<Zywiec>ORG ("Full Light" no anotado, ref. A.1.7)	<Zywiec>ORG <Full Light>MISC	<Zywiec>NAM.ORG (ref. 9.3.2)	I, C
Empire State Building	no anotado (ref. 4.2.3)	<Empire State>LOC	<Empire State Building> NAM.FAC.Building (ref. 9.3.2)	I, C, B
Alpine Skiing-Women 's World Cup Downhill	no anotado (ref. A.2.4)	<World Cup>MISC (ref. guía)	<Women>NOM <World>NOM (ref. 9.3.3)	I, C, B
the new upper house of Czech parliament	<parliament>ORG (ref. A.4.3, A.1.5)	<Czech>LOC	<Czech parliament>NOM (ref. 9.3.2)	I, C, B
Stalinist nations	no anotado (ref. A.1.6)	<Stalinist>MISC	no anotado (ref. 5.2.1)	I
Wall Street Journal	no anotado (ref. A.1.7)	<Wall Street Journal>ORG	<Wall Street Journal> NAM.ORG.MED (ref. 9.5.3)	I

Tabla 19. Relaciones aproximadas entre los tipos de entidad reconocidos en MUC7, CoNLL03 y ACE08.

MUC-7	CoNLL-03	ACE-08
Persona	Persona	Persona (excluídos nombres de animales)
Organismo	Organismo	Organismo
Localization	Localization	Localización / Localización geo-política/ Instalación
–	Miscelánea	Persona (no como grupo) / Vehículo / Arma
Hora / Fecha	–	Expresión temporal (tarea independiente)
Moneda / Porcentaje	–	Cantidad (tarea independiente)

4.4.2. Criterios para su identificación y anotación

Existe cierta confusión en torno a la anotación de entidades de nombre en los foros tradicionales que se aprecia en las propias guías de anotación. Así, un mismo elemento puede ser entidad o no, o ser clasificada siguiendo diferentes criterios según la guía concreta que se use. Por ejemplo, la guía de la ACE establece que en muchas situaciones resulta confuso diferenciar entre nombres de entidades y otras menciones (referencias a entidades). Por ejemplo, *health ministry* es considerada una referencia (*nominal mention*), pero *ministry of health* es considerada entidad (*name mention*). Y esto es debido exclusivamente al número de palabras, siguiendo una llamada *trumping rule*. Otro ejemplo es que los nombres de santos no son etiquetados en MUC *because removal of a saint's title leaves a non-unique name*, aunque los nombres de presidentes sí son etiquetados y eliminar los títulos en estos casos sí es correcto. No está claro cómo proceder en casos de metonimia (el uso de un término en lugar de otro concepto relacionado, como por ejemplo al utilizar "La Corona" para referise a la monarquía). La guía de MUC establece que cuando es común, la semántica del término ha de ser mantenida. De acuerdo a esto resulta que en la frase "Baltimore venció a los Yankees", Baltimore es etiquetado como localización y Yankees como equipo deportivo (categoría contemplada como parte de organismo tanto en CoNLL como en la ACE). En otros casos sin embargo, la semántica del texto es mantenida: los nombres de aeropuertos son etiquetados como organismos y no como localizaciones (cuando coincide con el nombre de una localización) si se refieren al aeropuerto como organismo o negocio. Los criterios de anotación de la ACE siempre adoptan esta postura, y tratan de mantener la intención del autor en relación con el texto.

4.4.3. Límites válidos

Los límites de las entidades son diferentes dependiendo del foro. Por ejemplo, en los corpus de MUC y CoNLL es común encontrar etiquetados sólo los elementos principales de un sintagma nominal, ignorando palabras que puedan estar asociadas como parte de ese sintagma (e.g. *Vice President*<*John Hime*>PER). Incluso cuando la palabra o palabras principales funcionan como cualificadores, pueden ser etiquetadas como NE, ignorando el núcleo del sintagma (e.g. <*Bridgestone*>ORG *profits*, <*Clinton*>PER *government*, <*Kennedy*>PER *family*). Es incluso posible encontrar palabras compuestas que han sido etiquetadas parcialmente (e.g. <*Ford*>ORG *Taurus*), argumentando que esto puede hacerse cuando está claro por el contexto o el conocimiento del anotador que esa parte del nombre compuesto es una entidad. En ACE los sintagmas nominales y su categoría semántica es respetada como un todo, y las entidades que pueda contener son anotadas como etiquetas anidadas (Tabla 20).

Tabla 20. Ejemplo de etiquetado de la frase *new york's la guardia airport […]* de acuerdo a la ACE. NOM: nominal, NAM: name, FAC: facility, GPE: geo-political entity, PopCenter: population centre, SPC: specific referent.

new york's la guardia airport	
<new york's la guardia airport>	NAM.FAC.AIRPORT.SPC
<new york>	NAM.GPE.PopCenter.SPC

Estos conflictos de criterios de anotación y límites se traducen en evaluaciones no comparables entre sí. A esto se une que los resultados son evaluados de modo diferente: en MUC, la identificación únicamente es considerada válida si los límites de la entidad son válidos, mientras que la clasificación se considera válida cuando el tipo asignado sea el correcto aunque los límites no sean exactos (mientras se solapen). En CoNLL03 se considera válido un resultado únicamente si coincide en el tipo y en los límites. En la ACE las identificaciones parciales son tenidas en cuenta únicamente si la parte principal de la entidad es reconocida al menos en una cierta proporción de caracteres.

4.5. Implicaciones en las herramientas NER

La ambigüedad existente sobre el concepto de entidad de nombre afecta también a las herramientas. En el Capítulo 3 analizamos siete herramientas NER genéricas, tanto de investigación (Annie, Afner, TextPro, YooName y Supersense Tagger con los tres modelos de aprendizaje que incluye) como comerciales (ClearForest and LingPipe). Los tipos de NE reconocidos están generalmente predefinidos y son muy diferentes entre ellas. Aparentemente todas las herramientas coinciden en el reconocimiento de los tipos persona, organismo y localización, pero existen muchas discrepancias con el resto. Las fechas y las cantidades numéricas son reconocidas como entidades por Annie (fechas, cantidades monetarias y porcentajes), Afner (fechas) y YooName (cantidades monetarias, fechas y otras unidades de medida). Otras categorías, como productos alimenticios y elementos naturales, o incluso nombres de eventos como guerras y huracanes son mucho menos frecuentes y sólo YooName las reconoce (al menos sin requerir adaptación previa).

Un análisis realizado sobre un corpus de pequeño tamaño y temática general muestra que, además de discrepancias en cuanto a los límites de las entidades establecidos por cada herramienta, podemos pasar de reconocer al menos 13 tipos diferentes de semánticas (sin incluir subtipos) a tan sólo 3 tipos diferentes [Marrero et al., 2009a][Marrero et al., 2009b]. Como consecuencia, resulta difícil saber qué esperar de una herramienta NER genérica.

En ciertos casos estas herramientas son adaptables a nuevas entidades mediante aprendizaje con corpus anotados. Este es el caso de la mayoría de herramientas basadas en aprendizaje automático supervisado, donde el uso de diferentes corpus de entrenamiento resulta en la identificación de entidades completamente diferentes. El problema en este caso es precisamente la existencia de estos corpus, que de nuevo, viene sesgado por las competiciones realizadas. Además, si los sistemas no son independientes de corpus y necesitan etiquetados múltiples ejemplos para poder funcionar, entonces no tiene aplicación práctica para un usuario medio.

Es frecuente además encontrar categorías fácilmente reconocibles por rasgos tipográficos. Este es el caso de los números y fechas, que curiosamente no suelen ser

reconocidos cuando aparecen escritas de modo alfabético en lugar de con dígitos. En estos casos parece primar la facilidad en su reconocimiento frente a la utilidad. A esto se añade la existencia de categorías claramente ambiguas, como "miscelánea". YooName reconoce fechas, cantidades monetarias y otras medidas, pero otras herramientas no las reconocen en absoluto o las clasifican en una categoría llamada *miscellaneous, unknown* u *others*. Sin información adicional, esta categoría tiene una dudosa aplicación práctica, ya que implica que una entidad es útil aún sin saber cuál es su semántica.

4.6. ¿Está NER realmente resuelta?

En MUC se obtuvieron tasas muy elevadas de precision y recall, por encima del 90%. Concretamente el mejor sistema obtuvo un 96% de recall y un 97% en precision. En CoNLL03 Participaron un total de 16 sistemas, con una media cercana al 90% el mejor sistema, y de algo más del 60% el peor. En ACE 2005 se obtienen tasas entre el 4 y el 71'9%, con cinco de los diez sistemas participantes en torno al 70%. En ACE 2008 sin embargo, las mejores tasas son tan sólo ligeramente superiores al 50%. En ambos casos las medidas usadas son tan diferentes de las tradicionales precision-recall que no pueden ser comparadas con MUC y CoNLL. En cualquier caso, el hecho de que las mejores tasas sean sólo del 50% en 2008, podría indicar un importante hueco en efectividad que aún queda por cubrir. A pesar de ello, en 2005 se afirma que está resuelta, con tasas superiores al 95%, y tras ACE 2008, este tipo de tareas desaparecen del panorama internacional. ¿Pero es esta afirmación cierta? para que esta tarea pueda considerarse resuelta las evaluaciones han de cumplir los requisitos del método científico [Katzer et al., 1998][Mitchell & Jolley, 2009][Harman, 2011]. A continuación se analizarán algunos de los criterios de validez que toda evaluación ha de cumplir.

4.6.1. Validez de contenido

La validez de contenido evalúa hasta qué punto las unidades de evaluación representan los elementos del dominio bajo estudio [Urbano, 2011][Trochim & Donnelly, 2007][Cook & Campbell, 1979]. Este requisito está muy relacionado con que la tarea refleje las necesidades de los usuarios a los que se orienta. En este sentido, en NER la tarea debe contar con unas semánticas adecuadas a los requisitos de los usuarios, además de una colección acorde con lo esperable en el mundo real en la medida de lo posible.

Las semánticas de interés en NER provienen de sus aplicaciones. Éstas son múltiples, y entre ellas encontramos la población de ontologías, con herramientas como KnowItAll [Etzioni et al., 2005], capaz de identificar ejemplos similares a los introducidos a partir de técnicas de bootstrapping sobre la Web. También la reciente área de Minería de Opiniones es susceptible de usar técnicas de NER para la identificación de los productos o conceptos de interés (productos, instalaciones, etc.). De hecho, el sistema OPINE [Popescu & Etzioni, 2005] ha sido desarrollado para la extracción de atributos de productos y el análisis de las opiniones relacionadas, y está basado en la herramienta KnowItAll. Estas aplicaciones abren las puertas al reconocimiento de prácticamente cualquier concepto. Las herramientas en NER de ámbito general también han dado muestra de esta situación, y han pasado de reconocer unos pocos tipos de entidades, siempre procedentes del entorno periodístico (personas, localizaciones, organismos, fechas) o militar (vehículos, armas), a nuevas categorías como productos alimenticios y nombres de eventos como guerras o huracanes, como es el caso de YooName. Las herramientas comerciales cada vez ofrecen más tipos de entidades predefinidos, además de la posibilidad de adaptación a nuevas entidades requeridas por el cliente.

El creciente aumento de los tipos de entidades sobre los que se necesita aplicar herramientas NER contrasta con la limitación de tipos evaluados en los foros internacionales hasta el 2008. La prolongada restricción de los tipos de entidades evaluados hace dudosa la generalización de estos resultados a cualquier tipo de entidad, más aún si tenemos en cuenta que también ha existido una importante limitación en cuanto a los corpus utilizados.

4.6.2. Validez externa

La validez externa mide hasta qué punto los resultados de un experimento pueden ser generalizados a otras poblaciones y configuraciones [Urbano, 2011][Trochim & Donnelly, 2007][Cook & Campbell, 1979]. Este el punto más débil en Recuperación de Información en general [Voorhees, 2002], donde tratar de generalizar a cualquier consulta y sobre cualquier colección es inabarcable. En realidad, ya ha sido mostrado que unos resultados positivos en una conferencia no implica un buen rendimiento con otros corpus [Voorhees, 2002]. Pero precisamente para limitar este conocido sesgo se utilizan grandes colecciones y se varían las consultas en cada edición. La aproximación tradicionalmente seguida consiste en el uso de grandes corpus y la creación de diferentes conjuntos de topics cada año, permitiendo a los investigadores comparar sus sistemas con diferentes colecciones en lugar de sólo con una.

Concretamente en el ámbito del reconocimiento de entidades se han detectado variaciones en los resultados debidas a diferencias en la densidad de los tipos de entidades a capturar, diferencias en el modo de citar entidades (por ejemplo, una persona no es citada del mismo modo en el Wall Street Journal que en el New York Times), diferencias en la longitud de los documentos, etc. [Vilain et al., 2007]. Para limitar este sesgo es importante contar con corpus heterogéneos, capaces de recoger una importante muestra de los posibles escenarios, lo que suele traducirse en grandes corpus. Sin embargo en NER los corpus utilizados en los foros tradicionales son de dimensiones reducidas y procedentes en su mayoría de artículos periodísticos. El mayor y más variado corpus en las evaluaciones tradicionales es el usado en la ACE a partir de 2005, con unos 50k words y con otros géneros además de los artículos periodísticos (*Broadcast Conversation transcripts, Broadcast News transcripts, Conversational Telephone Speech transcripts, Meeting transcripts, Newswire, Usernet Newsgroup/Discussion Groups* y *Weblogs*).

De hecho, los análisis realizados en el Capítulo 3 apuntan la pérdida de eficacia de herramientas NER ante otros corpus [Marrero et al., 2009a][Marrero et al., 2009b], y ha sido demostrado que cambiar las fuentes, incluso sólo en el género y no en el dominio, lleva a importantes pérdidas de eficacia de entre un 20 y un 40%. [Poibeau & Kosseim, 2001]. Esto

reduciría las tasas del 95% a alrededor del 65%, situando a NER a niveles de otras tareas de Extracción de Información consideradas mucho más difíciles y en absoluto resueltas, como es el caso de Resolución de Escenarios (*Scenario Template Production).*

4.6.3. Validez de convergencia

La validez de convergencia evalúa hasta qué punto los resultados del experimento son congruentes con otros resultados, teóricos o experimentales, con los que deberían estar relacionados [Urbano, 2011][Trochim & Donnelly, 2007][Cook & Campbell, 1979]. Por ejemplo, la validez de las evaluaciones en recuperación de información en general puede cuestionarse porque los resultados obtenidos pueden variar dependiendo de las personas que realicen los juicios de relevancia. Las tasas de acuerdo entre anotadores o evaluadores (*inter-annotator agreement*) nos dan una idea de cuáles son las tasas que es posible alcanzar en la evaluación. Por ejemplo, si la tasa de acuerdo es del 100%, un sistema automático tendría como límite superior ese mismo porcentaje de acierto, pero si las tasas sobre un corpus son del 50%, entonces difícilmente podremos exigirle a un sistema automático que acierte en más de un 50% de los casos, puesto que no existe acuerdo entre los restantes. Ha de existir cierta convergencia entre los resultados esperables y estas tasas.

Los ratios de acuerdo entre evaluadores dependen de tres factores principales: el conocimiento previo de los evaluadores sobre el dominio, las semánticas a evaluar, y la colección utilizada. En el ámbito de biomedicina, se apuntan tasas de consenso muy variables, de entre el 75 y el 90% para la anotación de genes y proteínas [Gaizauskas et al., 2003]. Pero en el caso de la ACE, contamos con tasas de más de un 80%. Concretamente en 2002, las tasas de acuerdo para la tarea EDT (*Entity Detection and Tracking*) mostraron tasas de 86%, 88% en 2003. Estas tasas son además similares a los ratios mostrados en MUC [Doddington et al., 2004].

Los sistemas evaluados hasta 2008 en la ACE están lejos de alcanzar estas tasas de acierto, sin embargo en MUC se supera de forma generalizada el 90% de micro-averaged F-measure, llegando a tasas del 96%. Hasta donde sabemos, no existe ninguna tarea en

recuperación de información donde las tasas superen de forma generalizada al acuerdo entre humanos (por otro lado, conflictivo, según vimos como resultado del análisis de las guías de anotación), que únicamente parece explicable debido al *overfitting* sobre los corpus de entrenamiento. Esto nos lleva, como vimos en el punto anterior, a limitaciones en la validez externa de las evaluaciones.

4.6.4. Validez de conclusión

La validez de conclusión evalúa hasta qué punto las conclusiones extraídas de los resultados del experimento están justificadas [Urbano, 2011][Trochim & Donnelly, 2007][Cook & Campbell, 1979]. De hecho, cualquier problema de validez en las evaluaciones nos lleva a que las conclusiones derivadas no sean válidas. En el caso de NER, los problemas anteriormente comentados muestran limitaciones en la validez de las conclusiones en cada foro, y ponen en duda que la tarea esté realmente resuelta.

Pero además, cada uno de los tres foros utiliza criterios y métricas diferentes para determinar la eficacia de los sistemas, lo que complica comparar los resultados incluso asumiendo semánticas y corpus similares. Por sí sola, la ACE es la única con una trayectoria suficiente como para medir la evolución del área y extraer ciertas conclusiones, sin embargo, sus métricas propias hacen difícil para la comunidad extraer conclusiones acerca del estado del arte y de cómo las mejoras, de haberlas, se trasladan a entornos operacionales reales. Por tanto, creemos que no existe suficiente evidencia que soporte la afirmación de que NER está resuelta, sino más bien lo contrario.

4.7. Foros actuales

Las competiciones tradicionales han dejado paso a las nuevas competiciones XER y ET. Como ventaja, estos nuevos foros amplían notablemente el número de semánticas a capturar y la colección. Pero no sustituyen la evaluación de la tarea evaluada hasta 2008. En

estas nuevas competiciones el concepto de entidad se reduce al de unidad documental. De hecho, en ET se adoptó la definición de entidad como *something with a homepage* (http://ilps.science.uva.nl/trec-entity). Las unidades de recuperación están por tanto perfectamente definidas y es posible aplicar métodos similares a los que se aplican en recuperación de información para su evaluación, ampliamente estudiados en TREC. El problema aquí es que las técnicas utilizadas para localizar un documento no tienen por qué ser ni siquiera similares a las empleadas en la localización y delimitación de texto, necesarias en muchas aplicaciones de NER. Igual que las técnicas aplicadas a recuperación de documentos y a los sistemas de pregunta-respuesta son diferentes, aunque en este último las respuestas vayan soportadas por documentos.

Otro problema es que la medida de efectividad en las herramientas NER viene dada no sólo por la precisión, sino también por la exhaustividad. En una herramienta NER utilizada para anotación semántica o población de ontologías, por ejemplo, nos interesa capturar todos los elementos que cumplan una semántica concreta. Por el contrario, en un motor de búsqueda, ante una colección de considerables dimensiones, interesa la recuperación temprana de documentos relevantes más que la recuperación de todos los documentos relevantes. Como consecuencia, las medidas empleadas se centran en la precisión y ordenación, pero no en exhaustividad. Esto nos llevaría a un problema de validez de construcción, puesto que las variables del experimento (la evaluación en este caso) no corresponden al propósito [Urbano, 2011].

Por último, hay que tener en cuenta que no todas las potenciales semánticas objeto de interés tienen páginas web representativas (e.g. cantidades monetarias o fechas), y tampoco la tienen todas las posibles entidades de una misma semántica, por muy común que parezca (e.g. no todas las empresas tienen página web).

En definitiva, son necesarias técnicas y herramientas capaces de reconocer semánticas de interés en texto libre que no necesariamente requieran la existencia de documentos como páginas Web o entradas de Wikipedia en los que se sustenten. Por tanto, los nuevos foros relacionados con NER, no pueden ser considerados como los modernos MUC, CoNLL o

ACE, porque tratan tareas diferentes. Los resultados de estos nuevos foros han de relacionarse con la tarea tradicional de NER con precaución.

4.8. Retos y oportunidades en NER

El Reconocimiento de Entidades Nombradas no es una tarea resuelta, pero puede resolverse. Al menos, hasta donde otras tareas dependientes de dominio pueden llegar a considerarse resueltas. El problema es que las evaluaciones y recursos utilizados hasta el momento no nos permiten decidir hasta qué punto lo está. NER ha sido considerado un problema resuelto a partir de tasas de efectividad obtenidas con un pequeño conjunto de tipos de entidad y unos pocos géneros documentales, habitualmente sólo del dominio periodístico. No sabemos qué ocurre al aplicar las técnicas actuales a otros tipos de entidad y otras colecciones, y no existen recursos para la evaluación de los nuevos tipos de entidades que las herramientas actuales en el mercado reconocen. Los nuevos foros de evaluación, aunque superan algunas de las limitaciones previas, no son suficientes para medir la evolución de NER porque evalúan sistemas con diferentes objetivos, no válidos para buena parte de las aplicaciones del área.

La comunidad de NER tiene la oportunidad de avanzar en el reconocimiento de cualquier tipo de entidad sobre cualquier tipo de colección. Pero el desarrollo de tales técnicas y su evaluación requerirían un enorme esfuerzo. En lugar de esto, los foros de evaluación deberían centrarse en aplicaciones específicas que permitan a los investigadores enfocar los esfuerzos en necesidades concretas de usuario, y de este modo, progresivamente mejorar las técnicas NER en general. En particular, es necesario extender la tipología de entidades a evaluar, y variarla con cierta frecuencia, adaptándose a los perfiles actuales de usuario. Las colecciones de evaluación deben ser también ampliadas, prestando especial atención a la aplicación de NER sobre escenarios tan heterogéneos como la Web. Además, estos foros de evaluación han de mantenerse a lo largo del tiempo, con métricas estables y compartidas por la comunidad científica. De este modo podremos

medir el estado del arte para necesidades y dominios específicos y, en algún momento, disponer de suficiente información para determinar si la tarea de NER es un problema resuelto o no. Mientras, sólo podemos determinar esta cuestión para casos específicos.

Lograr evaluaciones adecuadas de NER dadas las características de la tarea y no incurrir en elevados costes es un reto considerable. Las medidas de eficacia ampliamente utilizadas en otras áreas no son directamente aplicables a NER, y la metodología de evaluación ha de ser reconsiderada. En particular, es necesario medir la exhaustividad en la recuperación, lo que puede resultar muy costoso al usar grandes colecciones. Una alternativa es localizar las entidades existentes con ayuda de herramientas NER, e incrementalmente validar y añadir las nuevas entidades identificadas por los sistemas. Aunque de este modo no es posible establecer una medida de exhaustividad exacta, la publicidad de la colección y los resultados haría posible comparar cualquier sistema con los evaluados y disminuye este problema. Este sistema ya fue utilizado en TREC-QA para el reconocimiento de respuestas en forma de listados de elementos (*QA-list task*), tarea que podría constituirse como referente en la evaluación de NER por su similitud.

Finalmente, es necesario reconsiderar el esfuerzo requerido para adaptar una herramienta a un nuevo tipo de entidad o colección. El uso recurrente de técnicas de aprendizaje supervisado durante la pasada década ha contribuido a hacer las herramientas portables, pero a costa de grandes esfuerzos en anotación por parte del usuario. Una herramienta que para identificar nuevos tipos de entidades requiere corpus etiquetados de varios cientos de megabytes no es, o no debería ser comparable, a otra que tan sólo requiere validar un puñado de ejemplos. La utilidad de la primera depende en gran medida de los recursos disponibles por el propio usuario y su evaluación debe contemplar este factor. Por tanto, las evaluaciones en reconocimiento de entidades no sólo deben incluir medidas de efectividad, sino también de coste, que indique cuánto esfuerzo requiere por parte del usuario. Esta consideración promovería la investigación en metodologías de bajo coste, capaces de adaptarse a nuevas entidades en nuevas colecciones, reduciendo el esfuerzo del usuario para ello. Las técnicas de bootstrapping y de aprendizaje activo son ejemplos en esta línea. Adicionalmente, el desarrollo de este tipo de herramientas facilitaría la anotación

de los recursos necesarios para realizar la evaluación dinámica y continuada en el tiempo que es necesaria para medir la evolución del área.

Por tanto, y a pesar del conflicto lingüístico sobre el término *Named Entity* y los problemas derivados en herramientas y foros de evaluación, avanzar en NER es factible. Pero como en otras áreas, este avance ha de ser progresivo. No es necesario construir herramientas NER genéricas, capaces de capturar cualquier cosa y sobre cualquier colección, sino avanzar en técnicas y recursos que prueben ser útiles en diferentes aplicaciones. Esto, eventualmente, conducirá a un significativo grado de portabilidad ante diferentes entidades y dominios de aplicación.

4.9. Conclusiones

El Reconocimiento de Entidades Nombradas juega un importante papel en la gestión de la información, y es base de otras tareas de Extracción de Información como la generación de relaciones y la composición de escenarios. Sin embargo, las definiciones del término *Named Entity* han sido diversas, confusas e incongruentes. La definición de NE según el propósito y dominio de aplicación parece ser la única consistente. Por tanto, consideraremos que una Entidad Nombrada es cualquier semántica que pueda considerarse de interés en un dominio de aplicación.

La evaluación de las herramientas NER ha sido llevada a cabo en diversos foros, y ha sido incluso considerada un problema resuelto, con elevadas tasas de efectividad. Pero estas evaluaciones han sido realizadas usando un limitado conjunto de entidades prácticamente invariable año tras año, y con corpus muy pequeños en relación a los utilizados en otras áreas de la recuperación de información. Ambos factores conducen al *overfitting* sobre colecciones particulares, limitan la evolución del área y llevan a conclusiones erróneas al generalizar los resultados. Es necesario que la comunidad científica retome el área y desarrolle adecuados foros de evaluación, con una definición clara de la tarea y el modelo de usuario, además del uso de métricas apropiadas y

metodologías estándar. Sólo haciendo esto podremos contemplar la posibilidad de que NER se convierta en un problema resuelto.

Capítulo 5
Modelado de patrones para el Reconocimiento de Entidades: gramáticas IEG

Essentially, all models are wrong, but some are useful –George E. P. Box

5.1. Introducción

Los sistemas actuales de Reconocimiento de Entidades Nombradas deben hacer frente a uno de los problemas de la Extracción de Información en general: el tipo de contenido a reconocer ha de ser definido previamente. Esto conlleva problemas de dependencia del dominio [Turmo & Ageno, 2006]. De hecho, ya desde 1998 se establecen dos criterios en esta línea que deben cumplir los sistemas de extracción de información en general [Freitag, 1998]:

- *Retargetability*: adaptarse a un nuevo dominio no debería requerir modificar el código, como mucho debería requerir la eliminación/incorporación de nuevos atributos (*feature engineering)*.

- *Generality:* debería ser posible trabajar sobre un amplio rango de dominios y géneros, desde artículos periodísticos hasta páginas Web.

Como vimos en el capítulo anterior, los sistemas NER, como parte de los sistemas de Extracción de Información y dada su aplicación sobre cualquier dominio y semántica de interés, deberían guiarse por los mismos criterios. Lograr sistemas NER adaptables supone entonces que éstos han de ser capaces de reconocer diferentes léxicos y estructuras gramaticales, y adaptarse a las características del lenguaje [Ciravegna & Wilks, 2003a].

Por otro lado, la posibilidad de comprender los patrones subyacentes ayudaría a lograr mayor transparencia, y así poder reutilizar parte de ellos o modificarlos ante pequeños cambios. Esto contribuiría a hacer sistemas más adaptables, al tiempo que se facilita su desarrollo y mantenimiento. Esto último está además en línea con uno de los retos de la IE [Chiticariu et al., 2010a]. Para ello es necesario contar con un lenguaje de representación de patrones legible y descrito con cierto grado de estandarización que facilite su comprensión, y en último término, su adopción.

Pero no todos los modelos de representación son igual de adaptables: el modelado mediante wrappers, por ejemplo, supone grandes ventajas en el procesamiento de documentos Web, pero también grandes limitaciones en el procesamiento de otro tipo de documentos menos estructurados. La adaptabilidad de otros patrones como los basados en *bag-of-words* es muy limitada dada la falta de potencia para el modelado de diferentes estructuras gramaticales. Los modelos resultantes de ciertos tipos de clasificadores no son comprensibles, como es el caso de las redes neuronales, y aunque lo fueran, no son fácilmente editables por la complejidad del modelo y la falta de estandarización en su sintaxis. En otros casos, la limitación viene dada por la falta de flexibilidad para expresar nuevos léxicos o atributos del lenguaje.

La solución propuesta en el presente trabajo es la adopción de gramáticas independientes del contexto ligeramente modificadas para facilitar la descripción de cualquier característica del lenguaje que pueda ser útil para su identificación. A la potencia y flexibilidad de este enfoque se une la adaptación del estándar *Speech Recognition Grammar Specification* para su representación en la Web. El uso de un formalismo potente en cuanto a

las estructuras que es capaz de reconocer, y ampliamente estudiado como son las gramáticas independientes de contexto, unido a un estándar de representación en XML, lo convierte en una solución potente, flexible, editable y con cierto grado de estandarización que facilita su adopción en el nuevo entorno de la IE. La descripción de esta aproximación y su impacto en la eficiencia y grado de estandarización respecto a las gramáticas independientes del contexto convencionales son objeto de este capítulo.

5.2. Características de los modelos de representación

Como vimos en la Sección 2.4, los modelos de representación usados en NER son muy variados. Un modo de medir su potencia o capacidad de expresión es siguiendo la Jerarquía de Chomsky. Noam Chomsky clasifica las gramáticas formales en cuatro niveles, donde las gramáticas de nivel superior pueden reconocer los lenguajes generados por las de nivel inferior:

- Gramáticas de tipo 0 (no restringidas): son las más potentes, y por tanto capaces de reconocer los lenguajes recursivamente enumerables, es decir, todos aquellos reconocidos por una máquina de Turing. Su complejidad es indecidible.
- Gramáticas de tipo 1 (dependientes del contexto): son capaces de reconocer los lenguajes sensibles al contexto. Requieren máquinas de Turing deterministas, con una complejidad exponencial respecto a la longitud de la cadena de entrada.
- Gramáticas de tipo 2 (libres de contexto): reconocen los lenguajes independientes del contexto. Pueden implementarse mediante autómatas a pila, con una complejidad polinomial respecto a la longitud de la cadena de entrada.
- Gramáticas de tipo 3 (regulares): reconocen los lenguajes regulares. Pueden implementarse con autómatas finitos, por lo que su complejidad es lineal respecto a la longitud de la cadena de entrada.

Los modelos utilizados en los métodos Ad-hoc o de bootstrapping habitualmente tienen la capacidad de expresión de los lenguajes regulares. Esto supone mayor potencia que por ejemplo los árboles de decisión, y otros modelos usados por los clasificadores (e.g. redes de neuronas, máquinas vectoriales, etc.), donde no es posible representar la clausura de Kleene (e.g. no es posible representar el lenguaje $a*$).

Como contrapartida, los métodos Ad-hoc requieren incorporar a sus patrones mecanismos para poder representar los diferentes atributos, ya que no es posible en una gramática formal tomar varios alfabetos de entrada a un mismo tiempo (e.g. caracteres, etiquetas morfo-sintácticas, pertenencia a un listado, etc.). La unificación de este tipo de patrones para alcanzar cierto grado de estandarización es una de las contribuciones del *Common Pattern Specification Language* (CPSL) [Appelt & Onyshkevych, 1998], que formaliza la representación de autómatas finitos en cascada, donde el lenguaje reconocido por el primer autómata genera etiquetas que se convertirán en el lenguaje de entrada del siguiente y así sucesivamente (Figura 9).

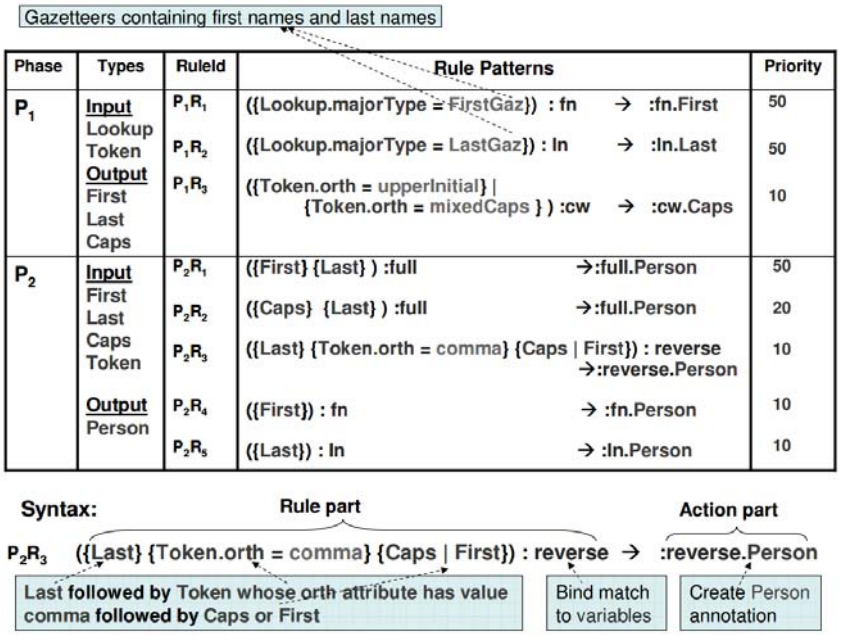


Figura 9. Ejemplo de autómata en cascada para el reconocimiento de nombres de persona [Chiticariu et al., 2010b]. Cada fase representa un autómata finito que reconoce como alfabeto de entrada los símbolos especificados en "Input". Las reglas contienen acciones (→) que indican los símbolos de salida, que serán los de entrada del autómata de la siguiente fase.

Este modelado basado en autómatas en cascada aporta mayor flexibilidad respecto a las gramáticas formales, y mayor grado de estandarización respecto a los modelos de métodos Ad-hoc. Sin embargo no está libre de limitaciones. Por ejemplo, al operar cada fase sobre una secuencia de anotaciones (o tokens) de izquierda a derecha, si existe más de una posibilidad de anotación, el motor ha de seleccionar una de ellas (generalmente adoptando una de tres opciones: la regla que consume más caracteres, la regla con mayor prioridad o la regla declarada antes en la gramática) sin posibilidad de ver las consecuencias en las siguientes fases. Si esta elección es errónea, el error se propagará a las siguientes fases (Figura 10a). La solución en JAPE (*Java Annotation Patterns Engine*), componente para la extracción de información del framework GATE [Cunningham et al., 2013], es permitir la existencia de múltiples reglas a la vez que pasan a la siguiente fase, donde pueden ser analizadas de forma conjunta (Figura 10b), con lo que dejan de actuar realmente como autómatas independientes en cascada. Esta y otras modificaciones a causa de las limitaciones de este modelo de representación (pueden verse en el trabajo de Chiticariu et al [2010]) lleva a adaptaciones, como las realizadas en JAPE o CAFETIERE [Rinaldi et al., 2005], que lo alejan del estándar CPSL, complicando su adopción.

El grado de estandarización es también un problema al usar clasificadores. A pesar de que el modelo en el que se basan los patrones es formal (e.g. árbol de decisión), no lo es su sintaxis, por lo que habitualmente se genera código específico de cada clasificador. Esto lleva a problemas de edición, puesto que, aún siendo comprensibles en algunos casos, resulta muy complejo realizar modificaciones sobre ellos. Una ventaja sin embargo es la flexibilidad que presentan ante el uso de nuevos atributos, ventaja que no suele estar presente al utilizar métodos Ad-hoc o de bootstrapping, dado que los atributos suelen estar predefinidos (e.g. RAPIER, WHISK, (LP)2).

Por último es necesario considerar aspectos de eficiencia: no todos los modelos de representación invierten los mismos recursos en la identificación del lenguaje que representan. Como hemos visto en la jerarquía de Chomsky, la eficiencia en el reconocimiento está reñida con la potencia del lenguaje. Este es un importante factor a considerar, ya que uno de los retos actuales es precisamente aplicar la extracción de

información a grandes colecciones de datos [Chiticariu et al., 2010a]. Para ello han aparecido lenguajes (e.g. AQL, xLog) que pueden acelerar la composición de escenarios en la identificación *multi-slot* frente al uso de expresiones regulares u otro tipo de gramáticas. Sin embargo, estas últimas continúan utilizándose en la captura de entidades y se trabaja en la reducción del tiempo de reconocimiento con mecanismos de optimización [Chiticariu et al., 2010b] e indización [Ramakrishnan et al., 2008].

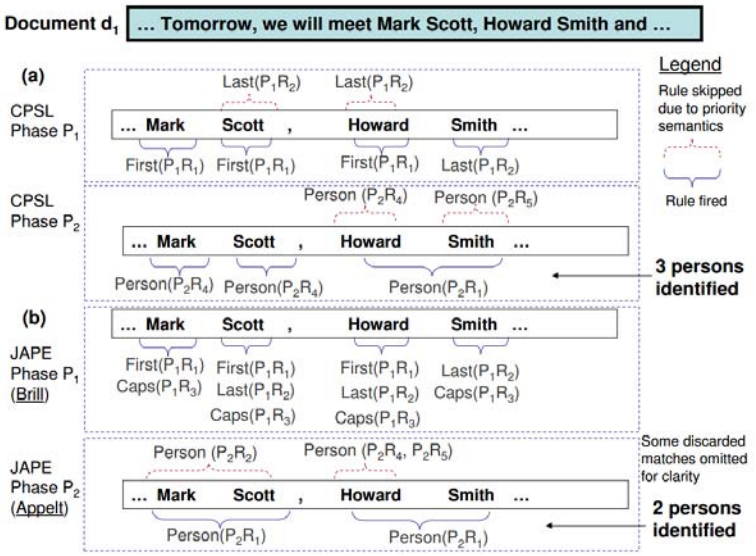


Figura 10. Ejemplo de salida de CPSL y JAPE [Chiticariu et al., 2010b]. Basado en la gramática de la Figura 9.

5.3. Gramáticas adaptadas a la Extracción de Información: IEG

El CPSL combina muchas de las ventajas de un modelo de representación ante documentos no estructurados, especialmente cuando no se conocen previamente los atributos a utilizar: potencia, cierta flexibilidad ante nuevos atributos, facilidad de edición y cierto grado de estandarización, puesto que no sólo está basado en las gramáticas formales sino que además ha sido formalizado.

Con la idea de utilizar un sistema de representación para las entidades (o elementos *single-slot*) con las ventajas del CPSL, pero eliminando sus inconvenientes, se ha adoptado

como modelo de representación gramáticas independientes de contexto con funciones asociadas. Esta aproximación mantiene además, como veremos, la complejidad computacional en tiempo en el reconocimiento de cadenas de entrada.

Para formalizar el modelo de representación propuesto comenzaremos describiendo las gramáticas independientes del contexto. Como ya se ha mencionado, estas gramáticas son un tipo de gramática formal (en concreto, de tipo 2), y sus componentes principales son [Hopcroft et al., 2006]:

- Un conjunto de símbolos terminales (Σ) que forma las cadenas del lenguaje que se está definiendo. Denominaremos a este conjunto "alfabeto".
- Un conjunto finito de símbolos no terminales ($\mathcal{V}$), denominados también variables o categorías sintácticas. Cada no terminal representa un lenguaje, es decir, un conjunto de cadenas.
- Un símbolo inicial ($S \in \mathcal{V}$), que es uno de los símbolos no terminales que representa el lenguaje que se está definiendo. Los restantes no terminales representan las cadenas auxiliares que se usan para definir el lenguaje del símbolo inicial.
- Un conjunto finito de reglas o producciones ($\mathcal{P}$) que representan la definición recursiva de un lenguaje. Cada producción consta de:
  - Símbolo no terminal al que define (parcialmente) la producción.
  - Símbolo de producción: →
  - Cadena formada por símbolos terminales y no terminales o bien por la cadena vacía (épsilon –ε). Esta cadena es llamada "cuerpo" de la producción.

Una gramática independiente del contexto se representa entonces como $G = (\mathcal{V}, \Sigma, \mathcal{P}, S)$. El lenguaje de G, designado como $L(G)$, es entonces el conjunto de cadenas terminales que tienen derivaciones desde el símbolo inicial. Es decir:

$$L(G) = \{w \in \Sigma^* | S \overset{*(G)}{\Longrightarrow} w\}$$

donde $\overset{*(G)}{\Longrightarrow}$ representa derivaciones en cero o más pasos de la gramática G, esto es, reemplazos de los no terminales por el cuerpo de una de sus producciones

consecutivamente desde el símbolo inicial hasta llegar a cadenas formadas exclusivamente por terminales.

Su uso en la Extracción de Información pasa por la posibilidad de expresar en ellas los atributos que caracterizan a las entidades a reconocer (ver Sección 2.3). Las gramáticas formales, expresadas en su forma original, únicamente soportan atributos relacionados con la sintaxis del alfabeto de entrada. No es posible por ejemplo indicar aspectos de formato o pertenencia a listados, a menos que modifiquemos cada vez el alfabeto de entrada, como ocurre con los transductores en cascada (e.g. CPSL). Para solucionar este problema hemos incluido funciones en los símbolos no terminales, de modo tal que para que una cadena de entrada $w \in \Sigma^*$ y un símbolo no terminal $A \in V \mid \mathrm{A} \overset{*(G)}{\Longrightarrow} w$, w será reconocida por el no terminal A sólo si cumple las funciones asociadas a A.

Formalizando, tenemos que una gramática independiente del contexto adaptada a IE (IEG) se define como $IEG = (G, \mathcal{F})$, donde $\mathcal{F}$ es un conjunto de funciones. Estas funciones toman como argumento una cadena $w \in \Sigma^*$ y devuelven un valor. Por ejemplo, una función g que tome como entrada una cadena de caracteres y devuelva su longitud se representaría como:

$$g: \Sigma^* \to \mathbb{N}$$

en general:

$$g: \Sigma^* \to Y_g$$

donde Y_g es el conjunto de posibles valores en el rango, y que depende del tipo de atributo y función en particular. Por ejemplo, en el caso de una función g que devuelva el lema de un término, el conjunto de posibles valores es M^*, siendo M el conjunto de letras del abecedario.

Podemos definir una función genérica f, utilizada para comprobar si las funciones g devuelven en cada caso el valor esperado dado una cadena cualquiera:

$$f: (\Sigma^* \to Y) \times Y \times \Sigma^* \to \mathbb{B}$$

$$\mathbb{B} = \{0,1\}$$

donde la función f cumple que:

$$f(g, y, w) = 1 \iff g(w) = y$$

Por tanto, toda condición que deba cumplir una cadena para ser reducida por un símbolo no terminal puede indicarse con una tupla $(\Sigma^* \to Y) \times Y$, incluyendo el atributo a evaluar sobre la cadena y el valor esperado del mismo. En el momento de reducir la cadena, la función f se evalúa por cada tupla, comprobando si todas las condiciones se cumplen o no. Definimos entonces cualquier derivación de una gramática IEG como:

$$A \overset{*(IEG)}{\Longrightarrow} w \;:=\; A \overset{*(G)}{\Longrightarrow} w \;\wedge\; \forall (g_i, y_i) \in F_A : f(g_i, y_i, w) = 1$$

Es decir, que es condición necesaria para todo lenguaje reconocido por A que todas y cada una de las funciones f_i asociadas a ese símbolo den como resultado 1 (true). Esto requiere almacenar por cada símbolo no terminal las tuplas con las funciones g y los valores esperados en cada caso. Por ejemplo, los nombres de persona representados en CPSL en la Figura 9 con una gramática en dos fases podrían expresarse mediante una única gramática IEG como la que aparece en el Código 9. En este ejemplo el símbolo inicial S reconocería todas las cadenas del alfabeto que según la gramática anteriormente descrita en dos fases, encajan en la definición de nombre de persona.

$$\mathrm{S} \to \mathrm{AL}|\mathrm{CL}|\mathrm{L}\,,\mathrm{A}|\mathrm{L},\mathrm{C}|\mathrm{A}|\mathrm{L}$$
$$\mathrm{A} \to \mathrm{T}$$
$$\mathrm{L} \to \mathrm{T}$$
$$\mathrm{C} \to \mathrm{D}|\mathrm{E}$$
$$\mathrm{D} \to \mathrm{T}$$
$$\mathrm{E} \to \mathrm{T}$$
$$\mathrm{T} \to [a - zA - Z0 - 9] +$$

$$\mathrm{F_A} = \{\mathrm{f(FirstGaz, true)}\}; \mathrm{F_L} = \{\mathrm{f(LastGaz, true)}\}$$
$$\mathrm{F_D} = \{\mathrm{f(UpperInitial, true)}\}; \mathrm{F_E} = \{\mathrm{f(mixedCaps, true)}\}$$

Código 9. Gramática IEG para la representación de nombres de persona tanto en modo directo como inverso[19].

Del mismo modo, podríamos representar por ejemplo números de teléfono que comiencen por 91, seguidos de dígitos o símbolos de puntuación, y que además estén en negrita como vemos en Código 10.

[19] La gramática podría reducirse permitiendo en el conjunto de funciones F el uso de operadores booleanos, con lo que en este caso podríamos prescindir del no terminal D incluyendo en F_C la función contenida en F_D mediante el operador lógico OR

En definitiva, combinamos de este modo la potencia sintáctica de las gramáticas independientes de contexto con la posibilidad de expresar en ellas diversas funciones (atributos) a un mismo tiempo y sobre cadenas de entrada de cualquier tamaño, eliminando así los inconvenientes de CPSL.

$$S \rightarrow 91AB$$
$$A \rightarrow [\backslash W \backslash d] *$$
$$B \rightarrow \backslash d$$
$$F_S = \{f(Strong, true)\}$$

Código 10. Gramática IEG para el reconocimiento de números que comienzan por 91 y están en negrita. La nomenclatura de expresiones regulares es la adoptada en el framework Microsoft DotNet[20].

5.4. Complejidad computacional de una gramática IEG

La incorporación de funciones en una gramática, ya sea regular mediante el uso de expresiones regulares, o independiente del contexto, tiene repercusiones en el tiempo necesario para el reconocimiento de cualquier cadena de texto. Pero esta complejidad depende en gran medida de la complejidad asociada a las funciones, más que de la incorporación de las funciones, como veremos a continuación.

5.4.1. Expresiones regulares

Si utilizamos expresiones regulares podemos realizar el reconocimiento con autómatas finitos. Dado que la unión, concatenación y clausura de un lenguaje regular resulta también en un lenguaje regular [Hopcroft et al., 2006], partiendo de una expresión regular puede construirse un autómata AFN-ε concatenando los autómatas correspondientes a estos operadores (Figura 11). Por tanto, si tenemos una expresión regular, aunque con funciones asociadas a ciertas sub-expresiones, podemos construir el autómata correspondiente a cada sub-expresión y concatenarlos entre sí siguiendo estas mismas reglas. Cada sub-expresión

[20] http://msdn.microsoft.com/es-es/library/system.text.regularexpressions(v=vs.80).aspx

de hecho puede convertirse a un AFD puesto que el lenguaje que reconoce es una expresión regular. La diferencia es que cada uno de estos autómatas, aunque reconozcan una cadena de entrada, no transitarán fuera de su estado o a un estado de aceptación si el resultado de las funciones asociadas sobre esa cadena es diferente a 1.

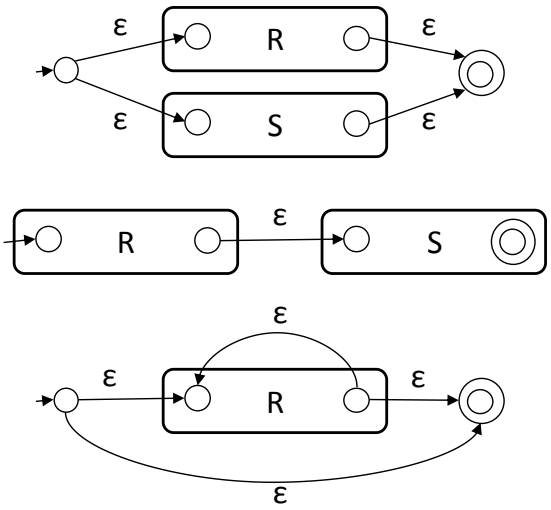


Figura 11. Construcción de un AFN-ε con un único estado de aceptación y sin arcos que entren al estado inicial o que salgan del estado de aceptación. Autómatas para la unión, concatenación y clausura. R y S son lenguajes regulares. El resultado, $L(R) \cup L(S)$, $L(R) \cap L(S)$, $L(R^*)$ es también un lenguaje regular [Hopcroft et al., 2006].

Una solución para implementar la comprobación de funciones comienza por la introducción de nuevas transiciones de tipo ε al entrar y salir de un autómata T con funciones asociadas, dejando fuera del autómata los estados de aceptación o estados finales (que no tienen por qué ser de aceptación). Los autómatas resultantes son equivalentes, y por tanto responderán a las operaciones de unión, concatenación y clausura del mismo modo (Figura 12).

Una vez tenemos los autómatas T con funciones asociadas representados de este modo, usaremos la transición ε de entrada para almacenar la posición en la cadena de entrada. De este modo en la transición ε de salida será posible aplicar la función o funciones a la cadena de entrada desde la posición de entrada hasta la posición actual. El autómata sólo continuará transitando si el resultado de esta operación es 1 (Figura 13). Estas transiciones sin embargo no serán consideradas para calcular la ε-closure, puesto que no se comportan realmente como transiciones ε.

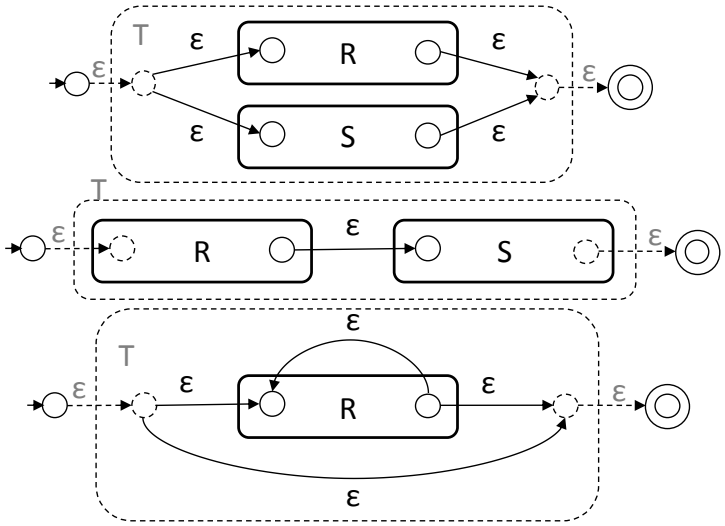


Figura 12. Autómatas equivalentes para la unión, concatenación y clausura con transiciones épsilon añadidas.

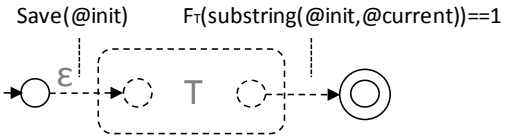


Figura 13. Autómata T con funciones asociadas. La transición ε de entrada provocará el almacenamiento de la posición en la cadena de entrada. El autómata sólo transitará al estado de aceptación (o a un nuevo autómata concatenado o que lo contenga) si se cumplen las funciones F_R sobre la cadena de entrada hasta la posición actual.

La complejidad en tiempo de un AFN-ε es $O(n)$, siendo n la longitud de la cadena de entrada. Para cada símbolo de la cadena de entrada, el autómata ha de calcular cuáles son los estados a los que transita y calcular la unión de los estados resultantes, lo que supone un tiempo de cómputo de s^2, siendo s el número de estados del autómata. A partir de ellos es necesario determinar los estados a los que se transita mediante transiciones ε, lo que supone de nuevo un tiempo s^2. Entonces el tiempo total de reconocimiento es $n \times (s^2 + s^2)$. Al incorporar las nuevas transiciones, añadiremos a este tiempo el necesario para determinar los nuevos estados a los que se llega a partir de ellas. Por cada símbolo de entrada, sumaremos el coste de las transiciones de entrada a los autómatas T, donde es necesario almacenar la posición en la cadena de entrada por cada uno de estos autómatas. Esto resultará en un tiempo $n \cdot (s \cdot c + s^2)$, siendo c el coste de almacenar la posición en la cadena de entrada, que se realizará en un máximo de s estados, y s^2 el coste de calcular las transiciones ε desde los nuevos estados. De forma similar, el coste de calcular las funciones

resultará $n \cdot (s \cdot d \cdot m + s^2)$, donde d es el tiempo de comprobación de un máximo de m funciones. Tenemos entonces que el tiempo de reconocimiento total es $n \cdot (s^2 + s^2 + (s \cdot c + s^2) + (s \cdot d \cdot m + s^2))$ o lo que es lo mismo $n \cdot (4s^2 + (s \cdot c) + (s \cdot d \cdot m))$.

Como resultado, el tiempo de reconocimiento se incrementa no sólo por la inclusión de estas nuevas transiciones (el número de estados como máximo se duplica), sino porque en el autómata resultante es más difícil reducir el número de estados: es fácil ver que dos autómatas T y T′ con funciones asociadas, aunque sean exactamente iguales, han de modelarse como autómatas diferentes. No sólo ocurre esto si las funciones F_T y F_T, son diferentes, sino que además, al tratarse de autómatas no deterministas, podemos estar al mismo tiempo en T y en T′ pero partiendo de diferentes puntos de entrada en la cadena a reconocer, lo que impide reducirlos a un único estado.

Sin embargo, es posible mantener la complejidad en tiempo de reconocimiento siempre y cuando el tiempo requerido en las operaciones de almacenamiento de la posición de la cadena de entrada y en la evaluación de las funciones sea constante. Esto último es factible utilizando mecanismos de indización previa de los las funciones sobre el texto. De este modo la complejidad puede continuar siendo lineal respecto al tamaño de la entrada.

5.4.2. Gramáticas independientes de contexto

Para cualquier gramática independiente de contexto (GIC) la comprobación de si una cadena de texto w pertenece al lenguaje que describe tiene una complejidad computacional en tiempo de $O(n^3)$ utilizando el algoritmo Cocke-Younger-Kasami (CYK) [Hopcroft et al., 2006]. Para saber si podemos mantener la misma complejidad computacional en una gramática IEG, primero debemos saber si es posible eliminar las producciones épsilon (cuerpo de la producción compuesto únicamente por la cadena vacía ε) y convertir la gramática a la Forma Normal de Chomsky, ambas condiciones necesarias para aplicar el algoritmo CYK.

Eliminación de producciones épsilon

Aunque el uso de producciones épsilon puede facilitar en muchos casos el diseño de gramáticas o autómatas, no son esenciales salvo para el lenguaje formado por la cadena vacía. Por tanto, si un lenguaje L tiene una GIC, entonces $L - \{\varepsilon\}$ tiene una GIC sin producciones épsilon. Si ε no pertenece a L entonces el propio L tiene una GIC sin producciones épsilon.

El algoritmo para esta transformación pasa por identificar los no terminales anulables. Un no terminal A es anulable si cumple que $A \overset{*}{\Rightarrow} \varepsilon$. Entonces cuando A aparece en el cuerpo de una producción, construimos dos versiones de la producción: una con el no terminal A y otra sin él, y eliminamos todas las producciones de la gramática cuyo cuerpo sea ε.

En el caso de gramáticas IEG tenemos que analizar qué ocurre si eliminamos no terminales con funciones asociadas cuyo cuerpo de producción sea épsilon, y si es posible sustituir el cuerpo de un no terminal con funciones asociadas por diferentes cuerpos de producción que eliminen los terminales anulables que contenga.

En el primer caso, en todo no terminal con una regla de producción épsilon podemos suprimir las funciones asociadas. El motivo es que dado que las funciones actúan sobre la cadena reconocida por el no terminal y ésta siempre es la cadena vacía, podríamos saber durante el diseño de la gramática si su valor es 0 ó 1. Si la función asociada a un no terminal A con producción épsilon da como resultado 1, entonces podemos seguir aplicando el algoritmo y por tanto eliminar la regla por ser anulable. Si por el contrario el resultado es 0, entonces nunca podrá reducirse por esa regla y por tanto deberá eliminarse y redefinirse la gramática en consecuencia.

Tampoco construir diferentes versiones de los cuerpos de producción que contengan no terminales anulables resulta ser un problema al tener funciones asociadas, puesto que la función o funciones se aplican sobre el resultado al reducir el no terminal, independientemente de cómo haya sido modelado su cuerpo de producción (Código 11).

```
Gramática G con producciones ε
S → AB
A → αAA|ε
B → bBB|ε
F_S = {…}; F_A = {…}; F_B = {…}

Gramática G' sin producciones ε
S → AB|A|B
~~S → ε~~
~~A → ε~~
A → αAA|αA|α
~~B → ε~~
B → bBB|bB|b
F_S = {…}; F_A = {…}; F_B = {…}
```

Código 11. Gramática G de tipo IEG con producciones épsilon y su transformación a G′ sin producciones épsilon: $L(G') = L(G) - \{\varepsilon\}$

Transformación a la Forma Normal de Chomsky

Una gramática se dice que está en la Forma Normal de Chomsky cuando cumple que todas sus producciones tienen una de las dos formas siguientes:

- A $\rightarrow BC$, donde A, B y C son no terminales ó
- A $\rightarrow \alpha$, donde A es un no terminal y α es un símbolo terminal

Si G es una GIC cuyo lenguaje consta de al menos una cadena distinta de épsilon, entonces existe una gramática G'' en la Forma Normal de Chomsky, tal que $L(G'') = L(G) - \{\varepsilon\}$. Como vimos, es posible eliminar las producciones épsilon de una GIC, obteniendo una gramática G', tal que $L(G') = L(G) - \{\varepsilon\}$. Para llegar a G'' a continuación es necesario:

- Conseguir que todos los cuerpos de longitud 2 o superior estén formados sólo por no terminales.
- Descomponer los cuerpos de no terminales de longitud 3 o superior en una cascada de producciones, teniendo cada una de ellas un cuerpo formado sólo por dos no terminales.

Para el primer caso tenemos que crear un nuevo no terminal para todo símbolo α que aparezca en un cuerpo de longitud 2 o superior, y utilizarlo en lugar del símbolo terminal en todos los cuerpos de producciones donde aparezca. Para el segundo caso, tenemos que descomponer las cadenas de producción de longitud superior a 3 en producciones con dos no terminales en cada cuerpo.

En ambos casos estamos sustituyendo parte del cuerpo de las producciones por nuevos no terminales. Como consecuencia, se agregan diferentes sublenguajes reconocidos por un no terminal S, en no terminales de su cuerpo de producción. Como las funciones aplican al lenguaje reconocido por S, y no a sus sublenguajes, podemos subdividir o componer éstos en cuantos sublenguajes queramos sin afectar al resultado. Por tanto podemos aplicar estas reglas a la gramática G' con funciones asociadas y obtener G'', que estará en la Forma Normal de Chomsky (Código 12).

```
Gramática G' sin producciones ε
S → αβγ F_S = {...}

Conseguir que todos los cuerpos de longitud ≥ 2 estén formados por no
terminales
S → ABC
A → α
B → β
C → γ  F_S = {...}

Conseguir que los cuerpos de longitud ≥ 3 estén formados sólo por dos
no terminales
S → RC
R → AB
A → α
B → β
C → γ  F_S = {...}
```

Código 12. Transformación de una gramática G' sin producciones épsilon, a una gramática G'' en la Forma Normal de Chomsky: $L(G'') = L(G')$

Aplicación de CYK

Una vez que tenemos la gramática en la Forma Normal de Chomsky, podemos aplicar el algoritmo CYK. Este algoritmo construye una tabla triangular, como se muestra en la Figura 14, donde se muestra un ejemplo para una cadena $w = a_1 a_2 \dots a_n$ con n=5. Cada una de las entradas de la tabla X_{ij} contiene aquellos no terminales capaces de derivar en uno o más pasos la cadena $w = a_i a_{i+1} \dots a_j$. La tabla se va llenando en sentido ascendente, de modo que en la primera línea irán aquellos no terminales que derivan directamente en cada terminal, en la línea superior se colocarán aquellos que se componen de terminales inferiores para formar la cadena de entrada correspondiente, y así sucesivamente. El algoritmo finaliza si en la posición superior existe algún no terminal capaz de generar la cadena de entrada completa. Podemos ver un ejemplo en la Figura 15.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

Figura 14. Tabla construida por el algoritmo CYK [Hopcroft et al., 2006].

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

{S,A,C}				
--	{S,A,C}			
--	{B}	{B}		
{S,A}	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	a	a	b	a

Figura 15. Ejemplo de gramática (izq.) y tabla correspondiente construída por el algoritmo CYK (dcha.) para la cadena de entrada w = baaba.

En el Código 13 se muestra el pseudocódigo del algoritmo. En él se consideran todas las posibles subsecuencias de una secuencia de palabras y establece P[i, j, k] a verdadero si la subsecuencia de palabras que empiezan desde i con longitud j puede ser generada desde los no terminales R_k. Una vez consideradas las subsecuencias de longitud 1, se continúa con las de longitud 2, y así sucesivamente. Para subsecuencias de longitud 2 o mayor, se considera cada posible partición de la subsecuencia en dos partes, y se comprueba si existe alguna regla P → Q R en la que Q concuerda con la primera parte y R con la segunda. Si es así, se establece que P concuerda con la subsecuencia completa. Una vez que este proceso se complete, la frase es reconocida por la gramática si la subsecuencia que contiene la frase completa concuerda con el símbolo inicial.

El algoritmo tiene una complejidad de $O(n^3)$ (con n igual a la longitud de la entrada), ya que se tarda un tiempo $O(n)$ en calcular cualquier entrada de la tabla y existen $n(n+1)/2$ entradas [Hopcroft et al., 2006]. Si los no terminales tienen asociadas funciones, se añadirá una nueva comprobación antes de igualar $P[i, j, R]$ a true, que consistirá en comprobar que

la subcadena que comienza en la posición i y finaliza en i+j-1, cumple las funciones asociadas al no terminal R (Código 14).

```
Let the input string consist of n letters, a1... an.
Let the grammar contain r terminal and nonterminal symbols R1... Rr.
This grammar contains the subset Rs which is the set of start symbols.
Let P[n,n,r] be an array of booleans. Initialize all elements of P to
false.
For each i = 1 to n
  For each unit production Rj → ai, set P[i,1,j] = true.
For each i = 2 to n -- Length of span
  For each j = 1 to n-i+1 -- Start of span
    For each k = 1 to i-1 -- Partition of span
      For each production RA -> RB RC
        If P[j,k,B] and P[j+k,i-k,C] then set P[j,i,A] = true
If any of P[1,n,x] is true (x is iterated over the set s, where s are
all the indices for Rs)
  Then string is member of language
  Else string is not member of language
```

Código 13. Algoritmo CYK para el reconocimiento de un lenguaje independiente de contexto.

La comprobación de estas funciones se añade al tiempo de generación de cada entrada de la tabla. Por tanto la complejidad de cálculo en cada entrada sería $O(n)$ multiplicada por el número de funciones y la complejidad de cada función. Si la complejidad de cada función es constante, continuaremos teniendo el mismo orden de complejidad. Por tanto si se da esta condición la complejidad total en el reconocimiento de cualquier cadena en una gramática IEG independiente de contexto continúa siendo $O(n^3)$.

```
Let the input string consist of n letters, a1... an.
Let the grammar contain r terminal and nonterminal symbols R1... Rr.
This grammar contains the subset Rs which is the set of start symbols.
Let P[n,n,r] be an array of booleans. Initialize all elements of P to
false.
For each i = 1 to n
  For each unit production Rj → ai
        If f(i)==1 for each f in Fj then
           set P[i,1,j] = true
For each i = 2 to n -- Length of span
  For each j = 1 to n-i+1 -- Start of span
    For each k = 1 to i-1 -- Partition of span
      For each production RA -> RB RC
        If P[j,k,B] and P[j+k,i-k,C] then
          If f(substring(j,i-1))==1 for each f in FA then
             set P[j,i,A] = true.
If any of P[1,n,x] is true (x is iterated over the set s, where s are
all the indices for Rs)
  Then string is member of language
  Else string is not member of language
```

Código 14. Algoritmo CYK para el reconocimiento de un lenguaje independiente de contexto con una gramática IEG. Los cambios introducidos respecto al algoritmo CYK aparecen en un recuadro.

5.5. Adaptación a *Speech Recognition Grammar Specification*

En la sintaxis de los modelos de representación es posible adoptar estándares, como es el caso de las expresiones regulares, o la notación BNF (*Backus-Naur Form*) para las gramáticas formales. Pero esto obligaría a introducir una notación específica para las funciones asociadas. Una alternativa que minimiza el impacto en aspectos de interoperabilidad, y además le confiere valor añadido, es la posibilidad de expresar estas gramáticas en lenguaje XML, mediante un estándar ya existente: el estándar SRGS (*Speech Recognition Grammar Specification*) [Hunt & McGlashan, 2004]. Esto haría de los patrones elementos fácilmente comprensibles, accesibles, distribuibles y gestionables (gracias a herramientas ya existentes), lo que promovería su uso. Estos aspectos son imprescindibles para reutilizar patrones o trozos de ellos, y dirigirnos a un modo de trabajo colaborativo.

El estándar SRGS especifica cómo mapear una gramática en formato ABNF (*Augmented Backus-Naur Form*) a XML mediante una DTD bien definida y aceptada[21]. El objetivo de este mapeo es el modelado de comandos de voz, de modo que sea posible guiar a los reconocedores de voz ante las órdenes de un usuario en la Web, pero la sintaxis es válida para representar cualquier gramática independiente de contexto en formato XML (Código 15).

```
<rule id="city">
   <one-of>
      <item>Boston</item>
      <item>"San Francisco"</item>
      <item>Madrid</item>
   </one-of>
</rule>
<rule id="command">
   <ruleref uri="#action"/>
   <ruleref uri="#object"/>
</rule>
```

Código 15. Ejemplo de gramática descrita en SRGS [Hunt & McGlashan, 2004].

A diferencia del estándar BNF, el ABNF permite la representación de expresiones regulares directamente gracias a los operadores de unión y repetición, y la posibilidad de

[21]http://www.w3.org/TR/speech-grammar/grammar.dtd

agrupar secuencias con paréntesis. Pero además su sintaxis mediante el estándar SRGS presenta ventajas añadidas frente a ABNF (Tabla 21):

- Especificación de pesos para reglas alternativas.
- Especificación de probabilidades en las repeticiones de un *item*.
- Posibilidad de expandir reglas con otras predefinidas como GARBAGE, NULL y VOID.
- Posibilidad de referenciar una gramática con su URI, de modo que sus reglas puedan ser usadas de modo unívoco por otras gramáticas
- Especificación de atributos de reglas tales como la visibilidad pública o privada (para indicar si puede o no ser usada por otra gramática) y atributos de gramática tales como la versión, la clase de alfabeto de entrada, la regla de inicio, etc.

Tabla 21. Mapeo de los principales elementos de ABNF (RFC5234) a SGRS.

	ABNF	**XML (SGRS)**
Rule definition	A = …	<grammar><rule id="A">…</rule></grammar>
Alternative	A = a / b A =/ c	<rule id="A"> <one-of><item>a</item>…</one-of></rule>
Alt. weight		<item weight="n">a</item>
Repetitions	<*min*>*<*max*>a <n>a	<item repeat=min-*max*>a</item>
Repetition probability		<item repeat=min-*max* repeat-prob="p">a</item>
Non-terminal reference	A = B C	<rule id="A"> <ruleref uri="gram#B"/>…</rule>

Para permitir la incorporación de funciones en el SRGS ha sido necesario añadir un nuevo elemento, que llamaremos *restriction,* dentro del elemento *rule*. De esta manera un no terminal (*rule)* puede contener cero o más restricciones, cada una apuntando a una función y el valor que ha de tomar (Código 16). Las referencias a funciones son URIs, que apuntan a procedimientos locales, pero también podrían apuntar a servicios Web, de modo que podemos crear repositorios distribuidos de funciones habitualmente usadas (e.g. negrita en documentos HMTL, localización en WordNet o Dbpedia, etc.) [Marrero et al., 2010a].

```
<!ENTITY % rule-expansion "#PCDATA | token | ruleref
| item | one-of | tag ">
<!ELEMENT rule (%rule-expansion; | example | restriction)*>
<!ATTLIST rule
    id ID #REQUIRED
    scope (private | public) "private">
<!ELEMENT example (#PCDATA)>
<!ELEMENT restriction EMPTY>
<!ATTLIST restriction uri CDATA #REQUIRED>
<!ATTLIST restriction value CDATA #REQUIRED>
```

Código 16. Extracto de DTD del estándar SRGS. En negrita las modificaciones introducidas para incorporar funciones.

La aplicación de este estándar a la Extracción de Información aporta ventajas también para los sistemas de anotación semántica, permitiendo referenciar de forma unívoca las gramáticas de las que procede cada anotación. Es posible además indicar la semántica descrita por la gramática mediante referencias a ontologías u otros recursos externos, por ejemplo incluyendo un atributo en cada gramática (o incluso en cada regla) que apunte a su URI. Esto es especialmente interesante en sistemas automáticos de anotación semántica, donde pueden modificarse todo un conjunto de anotaciones con pequeños cambios en los patrones de origen.

5.6. Conclusiones

El entorno en el que se enmarca el reconocimiento de entidades aspira a lenguajes estándares, que faciliten la construcción y mantenimiento de sistemas. En este sentido, es ventajoso contar con patrones legibles para el reconocimiento de entidades, que además puedan adaptarse a pequeños cambios en los corpus de aplicación sin necesidad de volver a generarlos de nuevo (con el coste que ello supone). La representación como gramáticas IEG y su sintaxis mediante el SRGS logra este objetivo. La potencia y flexibilidad de estas gramáticas están en línea además con la adaptabilidad requerida a cualquier sistema de Extracción de Información y por tanto, de Reconocimiento de Entidades.

Adicionalmente, el uso de estándares como base (gramáticas formales y XML-SRGS) favorece su adopción, almacenamiento y transferencia, permitiendo la creación de librerías de forma distribuida no sólo de los propios patrones o parte de ellos sino también de las

funciones utilizadas como atributos. Los cambios incorporados a estos estándares son mínimos y han sido formalizados, de modo que sea posible su incorporación en herramientas ya existentes o en nuevas herramientas a desarrollar.

Las gramáticas IEG no suponen mayor complejidad computacional que las gramáticas formales. Pueden adaptarse a los niveles de potencia requerida, modelándose como expresiones regulares, con una complejidad lineal, o bien como gramática independiente de contexto, con complejidad polinomial. Hay que tener en cuenta que incluso los lenguajes de marcado como HTML no pueden ser reconocidos mediante expresiones regulares, y la posibilidad de aplicar un mismo modelo de representación supone una ventaja en el camino hacia la estandarización. Para que esta complejidad se mantenga es necesario que las funciones asociadas tengan una complejidad constante. Esto es factible con indizaciones previas del texto.

Por último, la potencia y flexibilidad frente a los atributos que puede modelar convierten la combinación IEG-SRGS además en un modelo de representación apto para las herramientas de anotación semántica automática. Ofrece soporte a tipos documentales heterogéneos y la posibilidad de referenciar ontologías y relacionar así en una misma anotación los patrones de captura con las semánticas que identifica.

Capítulo 6
Aprendizaje inductivo de gramáticas IEG

Simpler methods may perform better when model assumptions are violated. And model assumptions are always violated, at least to some extent -John D. Cook

6.1. Introducción

Uno de los requisitos de los sistemas de reconocimiento de entidades es que han de ser tan portables como sea posible ante nuevas situaciones. La portabilidad de sistemas NER a diferentes tipos de documentos, diferentes dominios y distintas entidades pasa por lograr expresar adecuadamente en patrones de reconocimiento las estructuras sintácticas y los atributos que en cada caso permiten identificar las entidades de interés. Esto exige potencia en cuanto al lenguaje a reconocer, y flexibilidad en cuanto a los atributos a usar para ello. En el Capítulo 5 se describieron las gramáticas IEG y su representación mediante el lenguaje IEG-SRGS, que responden a estos requisitos, entre otros.

Pero además de la potencia respecto al lenguaje y flexibilidad respecto a los atributos, el usuario ha de jugar un papel importante en la adaptación de una herramienta. El desarrollo de patrones manuales es a menudo muy costoso y supone importantes conocimientos por

parte del usuario. Por ello es habitual el uso de métodos de generación automática o semi-automática de patrones, donde el usuario aporta su conocimiento a través de anotaciones (ver técnicas en Sección 2.2). En este escenario, es importante aprovechar el conocimiento que ya tiene el usuario acerca de las semánticas que desea reconocer, pero en términos de la cantidad de material anotado para entrenar las necesidades han de ser reducidas para minimizar la carga del lado del usuario [Ciravegna & Wilks, 2003a].

Como se ha visto en la Sección 2.2, dependiendo de la técnica utilizada es posible que se requiera del usuario desde unas pocas entidades a la anotación de grandes corpus. En el caso de los clasificadores es frecuente la necesidad de cientos o miles de anotaciones para obtener buenos resultados [Settles, 2010]. Pero estos corpus no siempre son accesibles o simplemente no existen para las entidades y dominios de interés. Y anotarlos es un proceso complejo y costoso por diversos motivos. En primer lugar el usuario ha de conocer las etiquetas y modo de anotación. En segundo lugar, ha de localizar ejemplos de entidades, lo que puede requerir revisar grandes cantidades de texto en corpus donde estas entidades aparecen de forma dispersa [Li et al., 2008a]. De hecho, las herramientas de anotación semántica se utilizan precisamente como ayuda en estas dos tareas descritas (ver Sección 2.5.2). En tercer lugar, la habilidad del usuario puede condicionar los resultados, no sólo por la capacidad para localizar ejemplos de entidades que respondan a patrones variados, sino también por la adecuada selección de los ejemplos negativos: la anotación de ejemplos de no entidades con características similares a las entidades es vital para localizar patrones precisos. Por último, en el caso de los clasificadores es necesario además que los ejemplos positivos y negativos estén equilibrados. Dado que lo habitual en este ámbito es que los ejemplos negativos sean muy superiores a los positivos, una alternativa habitual es eliminar parte de los ejemplos negativos o bien replicar los positivos. El problema es que la primera opción puede eliminar ejemplos negativos muy útiles, mientras que la segunda puede conducir a un overfitting sobre los ejemplos positivos replicados [Li et al., 2008a].

En este capítulo se diseñará un método para la generación semi-automática de gramáticas IEG para el reconocimiento de entidades. En primer lugar este método deberá ser flexible en cuanto a los atributos soportados, de modo que mantenga la flexibilidad de

las gramáticas IEG diseñadas para adaptarse a nuevos tipos de entidad. En segundo lugar, ha de evitar el uso de corpus anotados. Para lograr este objetivo se aprovechará el conocimiento del usuario, pero se le facilitará la tarea permitiéndole iniciar el proceso de generación de patrones tan sólo aportando ejemplos de entidades, que además no necesariamente han de estar en el corpus. Además se le guiará durante todo el proceso de anotación mediante la aplicación de técnicas de aprendizaje activo, donde será el sistema el que le propondrá las entidades que habrá de validar. Como resultado se podrán obtener expresiones regulares estándar o bien gramáticas IEG con la potencia de las expresiones regulares.

6.2. Trabajos relacionados

Generación de expresiones regulares

La tarea de reconocimiento de ciertas entidades puede llevarse a cabo mediante expresiones regulares cuando los patrones responden a secuencias de caracteres. Ejemplos de este tipo de entidades son las direcciones de correo electrónico, los números de teléfono, algunos nombres de genes y proteínas, etc. La creación manual de expresiones regulares es una solución ampliamente adoptada para la Extracción de Información [Li et al., 2008b]. Actualmente continúa teniendo mucha importancia, también en el entorno Web, donde existen múltiples repositorios de pequeños snippets de código, en su mayoría expresiones regulares hechas por los usuarios para la identificación de entidades específicas en determinados sitios Web, en un área que se ha dado a conocer como *Web Data Extraction o Web Scraping*.

El problema de inducir expresiones regulares a partir de ejemplos ha sido estudiado fuera del contexto de Extracción de Información, pero gran parte de este trabajo asume expresiones regulares simples y compactas, y el problema en Extracción de Información es que esto no siempre es así [Li et al., 2008b]. Otras entidades no pueden ser reconocidas mediante esta técnica porque requieren de gramáticas más potentes (e.g. lenguajes con

anidamientos). Además, lo habitual es encontrarnos con la existencia de múltiples alfabetos de entrada, cada uno correspondiente a un tipo de atributo diferente obtenido con anotadores morfo-sintácticos, listados, etc.

Esto se resuelve en los sistemas Ad-hoc mediante el uso de atributos predefinidos a nivel token. Es el caso de Whisk [Soderland, 1999], RAPIER [Califf & Mooney, 2004] o Amilcare [Ciravegna & Wilks, 2003b]. El primero utiliza información morfo-sintáctica, el segundo añade información semántica a partir de WordNet y Amilcare añade otros atributos como el lema, la categoría léxica e información sobre el uso de mayúsculas. Todos ellos generan patrones con la potencia de las expresiones regulares, pero no identifican patrones a nivel carácter. Además el resultado son patrones *ad-hoc*, lo que dificulta su adopción (Código 17).

```
Pattern:: * ( Person ) * ‘@Passive’ *F ‘named’ * {PP *F ( Position ) *
‘@succeed’ ( Person )
```

Código 17. Ejemplo de patrón generado por la herramienta Whisk. Person y Position son subexpresiones previamente almacenadas. El prefijo “@” indica el valor resultante de un stemmer. “*F” indica repetición de cero o más tokens pero con la misma categoría morfo-sintáctica.

Los clasificadores por el contrario sí cuentan con la flexibilidad suficiente para generar patrones de diferente granularidad, aunque no es habitual el uso de información acerca de la secuencia de caracteres más allá del uso de *n-grams* por la explosión combinatoria a la que pueden dar lugar.

Otra aproximación la encontramos en el trabajo de Brauer et. al [2011]. Este es el primer trabajo, según el conocimiento de sus autores, donde son analizados atributos de diferente granularidad (a nivel carácter y a nivel token) para inducir expresiones regulares para extracción de información. Concretamente, partiendo de ejemplos positivos, construyen autómatas a partir de los prefijos y sufijos de las entidades. Los atributos considerados pueden ser los propios caracteres (nivel instancia), clases de caracteres prefijadas (nivel clase) o bien tipos de tokens prefijados (nivel token). Estos alfabetos sin embargo pueden reducirse a uno solo, puesto que las clases de caracteres y de tokens se pueden construir como expresiones regulares sobre caracteres (e.g. un token numérico se expresa como [0-9]+).

También el trabajo de Wu y Pottenger [2005] utiliza atributos a nivel token y a nivel carácter. En este caso se identifican patrones sobre segmentos predefinidos de texto a partir de los propios tokens y su información morfo-sintáctica, pero el uso de atributos a nivel carácter se limita a la identificación de tokens numéricos.

El objetivo de este capítulo es la generación de patrones para el reconocimiento de entidades mediante el análisis de atributos tanto a nivel carácter como a nivel token. El uso de ambos tipos de atributos conducirá a la generación de gramáticas de tipo IEG, mientras que el uso únicamente de caracteres para la identificación de patrones dará lugar a la generación de expresiones regulares estándar.

Aprendizaje activo

El aprendizaje activo consiste en aportar ejemplos etiquetados durante el proceso de entrenamiento para la generación de patrones (ver Sección 2.2.2). Se aplica con el objetivo de reducir costes de anotación mediante la selección de los fragmentos de texto más relevantes a anotar. Se trata de una técnica útil en muchos problemas de aprendizaje automático donde los corpus no anotados son abundantes pero las anotaciones son dispersas o costosas de obtener [Settles, 2010]. Una consecuencia adicional de esta técnica es que permite obtener realimentación del usuario, de modo que es posible aportar nueva información con la que corregir la generación de patrones durante todo el proceso, y no ceñirse a la información inicial como ocurre en aprendizaje automático supervisado con grandes corpus, o en bootstrapping con un conjunto de semillas. De hecho en bootstrapping la falta de realimentación suele conducir a la rápida degradación de los resultados (ver Sección 2.2.2).

Al aplicarlo sobre clasificadores es habitual contar con un corpus de entrenamiento inicial formado por cierta cantidad de documentos anotados. En el área de NER en concreto esto puede verse tanto al usar clasificadores [Hachey et al., 2005] [Shen et al., 2004] como al usar sistemas Ad-hoc [Ciravegna & Wilks, 2003a][Thompson et al., 1999][Soderland, 1999]. Pero no sólo es necesario contar con una cierta cantidad de anotaciones iniciales, sino que además se ha observado que usar como unidad de anotación (texto ofrecido al usuario para

anotar) fragmentos de texto en lugar de documentos completos produce una caída muy significativa de la efectividad [Li et al., 2008a]. El problema de usar documentos completos como unidad de anotación es que los costes de anotación y resultados logrados pueden variar en función de las características documentales del corpus. También puede derivar en un conjunto de anotaciones positivas y negativas poco equilibrado, puesto que el número de ejemplos negativos es habitualmente muy superior al anotar un documento completo. Por otro lado, si los documentos en el corpus son de grandes dimensiones, los problemas de anotación serán similares a los de anotación de corpus descritos anteriormente. A esto se suman posibles problemas de permisos y seguridad para acceder a documentos completos.

Sin embargo, salvo el trabajo de Li et al., no conocemos otros trabajos en reconocimiento de entidades que no utilicen documentos anotados, a excepción del uso de tokens en el trabajo de Jones [2005], que utiliza bootstrapping. Sí es posible sin embargo encontrar trabajos que usan como unidad de anotación fragmentos de texto predefinidos, pero esto supone un conocimiento previo de los límites de las entidades. Este es el caso por ejemplo de algunos métodos Ad-hoc que incorporan aprendizaje activo como Whisk [Thompson et al., 1999] y el trabajo de Wu y Pottenger [2005].

En nuestro caso, dado que para facilitar la labor del usuario se utilizarán inicialmente únicamente un conjunto de entidades, se aplicarán técnicas de aprendizaje activo durante el proceso que soliciten del usuario la información necesaria. Pero a diferencia de la mayoría de las aproximaciones, se considerará como unidades de anotación fragmentos de texto no prefijados identificados por el sistema como potenciales entidades. De este modo se minimiza la cantidad de texto a anotar hasta el punto de transformarlo en una tarea de validación, en lugar de anotación, y se evita cualquier dependencia de las características de las entidades o de los documentos del corpus utilizado.

6.3. Generación de gramáticas IEG: descripción general del método

El problema al que nos enfrentamos puede ser formalizado como sigue. Dado un conjunto de ejemplos positivos o entidades $\mathcal{P}$ asumimos que existe una gramática G de tipo IEG capaz de generarlos, es decir, $\mathcal{P} \subseteq L(G)$. Esta gramática ha de generar el lenguaje del tipo de entidad concreto que buscamos, y no otro. Por tanto, si el conjunto de entidades potenciales es $\mathcal{A}$ tenemos que $L(G) = \mathcal{A}, \mathcal{P} \subseteq \mathcal{A}$. El objetivo de nuestro método será aproximar G mediante una gramática IEG G' regular o de tipo tres tal que se cumpla que $L(G') \approx L(G)$. Esta aproximación será medida en términos de precisión y exhaustividad.

Dado que debe ser posible la incorporación de cualquier atributo a nivel token, el usuario incorporará también las funciones correspondientes (función atributo). El sistema de forma automática añadirá además funciones de granularidad carácter.

En el esquema de la Figura 16 puede observarse el método a grandes rasgos. Para cada atributo, la función atributo es aplicada sobre las entidades conocidas y su contexto. Sobre los valores resultantes es aplicado un algoritmo para la identificación de patrones, que será detallado en la sección 6.6. Este algoritmo busca patrones en las entidades y su contexto, y devuelve reglas candidatas. Cada regla consta de un elemento (carácter o token) que ha de aparecer siempre en esa posición para cada entidad analizada. El proceso de AL se utiliza entonces para determinar si estas reglas son válidas o no. Para ello se identificarán entidades candidatas en el corpus no anotado que no respondan a las reglas creadas (y sí a las restantes ya creadas previamente), además de entidades similares para aumentar la exhaustividad, que el usuario deberá validar. Si las primeras resultan ser entidades, las reglas candidatas no serán creadas. Este proceso se realiza mientras existan reglas candidatas.

Los atributos de granularidad carácter introducidos de forma automática serán procesados en primer lugar (fase I). Se utilizan tres atributos diferentes, de más específico a más general, similares a los utilizados en Brauer et al. [Brauer et al., 2011]. Posteriormente serán procesados cada uno de los atributos a nivel token introducidos por el usuario (fase

II). Como resultado del procesamiento de los atributos a nivel carácter se obtendrán expresiones regulares, mientras que como resultado del procesamiento de los atributos a nivel token se establecerán condiciones sobre ciertos tokens en las expresiones regulares ya generadas. Por ejemplo, podemos utilizar como atributo a nivel token las etiquetas sintácticas de los tokens. Como resultado podríamos obtener expresiones regulares donde ciertos tokens han de responder a determinadas etiquetas sintácticas.

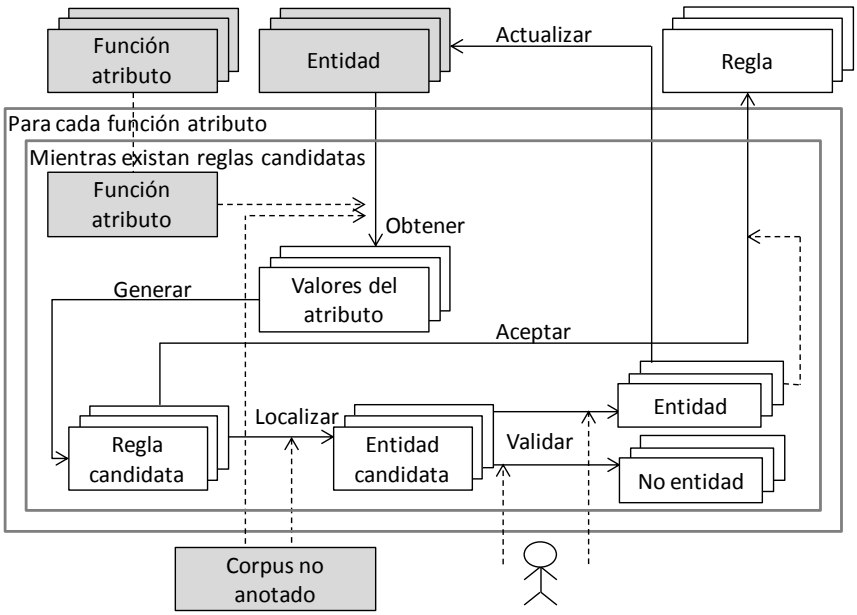


Figura 16. Método de generación de gramáticas IEG.

Las reglas generadas se representan como una gramática IEG que tomará como alfabeto de entrada valores de los atributos en la fase I, y como valores de las funciones asociadas a los no terminales los valores de los atributos de la fase II. Denotaremos el primer conjunto de valores como Σ_c y el segundo como Σ_t. Por tanto, si tenemos un conjunto de atributos a nivel token $\mathcal{G} = \{g_1, g_2, \dots, g_n\}$, el conjunto de posibles valores de todos los atributos considerados se definirá como:

$$\Sigma_t = \bigcup_{g \in \mathcal{G}} Y_g$$

De este modo, siguiendo las reglas establecidas en el Capítulo 5, el lenguaje de la gramática que queremos construir cumplirá:

$$L(G') = \left\{ w \in \Sigma_c^* \middle| S \overset{*}{\Rightarrow} w \right\}$$

donde S es el símbolo inicial. Y toda derivación desde un no terminal A a un token $\alpha \in w$ deberá cumplir:

$$A \overset{*(IEG)}{\Longrightarrow} \alpha \;:=\; A \overset{*}{\Rightarrow} \alpha \;\wedge\; \forall (g_i, y_i) \in F_A : f(g_i, y_i, \alpha) = 1$$

es decir, para todo no terminal A con una regla de producción involucrada en $S \overset{*}{\Rightarrow} \alpha$, α deberá cumplir las funciones f asociadas a A. Esto es, para cada par (g_i, f_i) asociado al no terminal A el atributo g_i sobre α deberá tomar el valor y_i.

6.3.1. Atributos a nivel carácter

Los atributos a nivel carácter se dividen en tres tipos: carácter, Unicode y agrupación-Unicode:

- Carácter (Σ_{c1}): se toman los caracteres correspondientes al texto de entrada. Estos caracteres son representados con su código Unicode.
- Unicode (Σ_{c2}): los caracteres son convertidos a categorías Unicode (Tabla 22). Estas categorías son definidas como una propiedad de los caracteres Unicode que sirve como clasificación básica del carácter según su uso principal [The Unicode Consortium, 2012]. La propiedad subdivide los caracteres en letras, dígitos, signos de puntuación y símbolos, entre otros, que a su vez se subdividen. Todas las categorías están representadas por dos letras, donde la primera letra da información acerca de la clase principal y la segunda indica la subclase dentro de ella. La representación como expresiones regulares de estos símbolos viene soportada por la mayor parte de los motores de expresiones regulares. En nuestro caso se ha utilizado el motor de expresiones regulares del Framework Microsoft .NET[22], donde la representación de una categoría Unicode se realiza con el prefijo $\backslash p$ seguido de las dos letras que la identifican. Se ha hecho una salvedad en cuanto a los símbolos que identifican a las letras en este sub-alfabeto. Dado que la variabilidad de éstas es muy grande, en lugar de considerar un carácter cualquiera como una letra mayúscula (Lu) o minúscula (Ll), continúa

[22] http://msdn.microsoft.com/es-es/library/system.text.regularexpressions(v=vs.80).aspx

manteniéndose el carácter concreto pero independiente de mayúscula. Por ejemplo, mientras que un dígito por ejemplo pasará a ser representado como $\backslash pNd$, la letra a pasará a ser representada como una expresión regular del tipo $(A \mid a)$, utilizando la codificación Unicode para representar cada una de las letras.

Tabla 22. Categorías Generales de Unicode

Lu	Letter, uppercase	**Pc**	Punctuation, connector
Ll	Letter, lowercase	**Pd**	Punctuation, dash
Lt	Letter, titlecase	**Ps**	Punctuation, open
Lm	Letter, modifier	**Pe**	Punctuation, close
Lo	Letter, other	**Pi**	Punctuation, initial
Mn	Mark, nonspacing	**Pf**	Punctuation, final quote
Mc	Mark,spacing	**Po**	Punctuation, other
Me	Mark, enclosing	**Zs**	Separator, space
Nd	Number, decimal digit	**Zl**	Separator, line
Nl	Number, letter	**Zp**	Separator, paragraph
No	Number, other	**Cc**	Other, control
Sm	Symbol, math	**Cf**	Other, format
Sc	Symbol, currency	**Cs**	Other, surrogate
Sk	Symbol, modifier	**Co**	Other, private use
So	Symbol, other	**Cn**	Other, not assigned

- Agrupación Unicode (Σ_{c3}): esta clase está formada por la agrupación de las categorías Unicode en torno a siete categorías propias: letra mayúscula, letra minúscula, símbolo, puntuación, separador y otros (Tabla 23). Esta representación es similar a la realizada en Brauer et al. [2011], con la salvedad de que en nuestro caso se diferencia entre símbolos y caracteres de puntuación, y además en el conjunto de separadores se incluyen los caracteres de control (e.g. salto de línea) en lugar de considerar únicamente los espacios en blanco. La representación de este sub-alfabeto se realiza con expresiones regulares formadas por la unión de las categorías correspondientes (e.g. $Control = (\backslash pCc|\backslash pZs)$).

Tabla 23. Agrupación propia de categorías Unicode

U: Letra mayúscula	Lu
L: Letra minúscula	Ll
D: Dígito	Nd
Y: Símbolo	Sm, Sc, So
S: Separador	Cc, Zs
P: Puntuación	Pc, Pd, Ps, Pe, Pi, Pf, Po
O: Otros	{restantes categorías Unicode}

6.3.2. Atributos a nivel token

Se considera un token como un conjunto de caracteres alfanuméricos delimitados por otros que no lo sean, o por inicio o fin del texto. Cada uno de los tokens de un texto de entrada puede tener cero o varios atributos asociados (lema, longitud, pertenencia a un listado, etc.). Como hemos visto, cada atributo vendrá representado por una función g que toma como entrada un token y da como salida un valor $y \in Y_g$, tal que $Y_g \subseteq \Sigma_t$. Las condiciones asociadas a no terminales que representan tokens en la gramática IEG son entonces definidas como tuplas (g, y).

Como vimos en la Sección 5.3, estas funciones pueden incorporarse en cualquier no terminal de la gramática IEG, y su cumplimiento es necesario para que puedan producirse las derivaciones. En este caso se restringe la potencia de estas gramáticas al permitir funciones tan sólo en no terminales que corresponden a tokens, y no a cualquier fragmento de texto. La complejidad en tiempo para comprobar el cumplimiento de las condiciones será constante ($O(1)$) una vez preprocesado el texto con todos los atributos de interés g y almacenados los valores resultantes para cada token. Como vimos en la Sección 5.4, dada esta condición, las gramáticas IEG no tienen mayor complejidad computacional en tiempo que las gramáticas formales. Por tanto la gramática G', que es regular, tendrá una complejidad lineal sobre el texto de entrada, como ocurre con cualquier expresión regular.

6.4. Fase I: reducción de caracteres

La metodología para la generación de gramáticas G' parte de un conjunto inicial de semillas positivas $\mathcal{I}$ y un corpus de entrada. Las semillas forman parte del conjunto $\mathcal{A}$ de entidades potenciales que queremos reconocer, pero no es necesario que se encuentren en el corpus de entrada, por lo que pueden ser obtenidas de bases de datos o listados. Durante la generación de la gramática G' el conjunto $\mathcal{Q}$ de ejemplos será incrementado mediante un proceso de aprendizaje activo. Este conjunto, que inicialmente sólo tendrá las semillas $\mathcal{I}$,

pasará entonces a contener el conjunto de entidades válidas $\mathcal{P}$ así como el de falsos positivos $\mathcal{N}$, de tal modo que:

$$\mathcal{I} \subseteq \mathcal{P}, \qquad \mathcal{P} \subseteq \mathcal{A}, \qquad \mathcal{N} \cap \mathcal{A} = \emptyset\,, \qquad \mathcal{Q} = \mathcal{P} \cup \mathcal{N}$$

El proceso se inicia con la construcción de una gramática IEG a partir de las semillas $\mathcal{I}$ aportadas por el usuario. El resultado será la gramática objetivo G', a la que se le aplicará iterativamente un algoritmo de inducción de reglas que fijará nuevos no terminales e incorporará funciones asociadas a ellos. Inicialmente esta gramática contiene un no terminal R_p por cada semilla, de tal modo que, además del símbolo inicial, existirán tantos no terminales como semillas. El cuerpo de producción de cada uno de ellos estará formado por la propia entidad, además del carácter anterior y posterior si los hubiera[23], representado todo ello con el alfabeto Σ_{c1} (Código 18).

$\mathcal{I} = \{86377535B, 26455535C, 94768235H\}$

$S \rightarrow R_{p1}|R_{p2}|R_{p3}$
$R_{p1} \rightarrow 86377535B$
$R_{p2} \rightarrow 26455535C$
$R_{p3} \rightarrow 94768235H$

Código 18. Gramática G' inicial generada a partir de las semillas. Su símbolo inicial es S, mientras que su alfabeto es Σ_{c1}.

En este punto se inicia la fase de análisis de atributos a nivel carácter. Se aplica el algoritmo de inducción que genera nuevas reglas a modo de no terminales R_c, reduciendo el número de terminales de la gramática (proceso de compresión). Cuando no pueden generarse más reglas de este tipo con este sub-alfabeto se pasa al siguiente, en este caso Σ_{c2}. Todos los terminales en los cuerpos de producción son transformados a este sub-alfabeto, a excepción de los contenidos en las reglas de inducción generadas en fases anteriores, que permanecerán sin cambios una vez creadas. Se repite el proceso para Σ_{c2} y posteriormente para Σ_{c3}, antes de pasar a la fase de análisis de atributos a nivel token. Hay que tener en cuenta además que cada vez que se aplica el algoritmo de inducción se usan además técnicas de aprendizaje activo, por lo que nuevos ejemplos positivos y negativos son incorporados al conjunto $\mathcal{Q}$, y los positivos además son incorporados a la gramática G' en

[23] En esta fase se limita el contexto de cada entidad a los caracteres inmediatamente anterior y posterior para evitar posible overfitting al tratarse de un atributo de granularidad muy fina.

forma de nuevos no terminales R_p (Código 19). Finalizado este proceso, aquellos caracteres alfabéticos sobre los que no se haya generado regla se mantienen con su valor original en el alfabeto Σ_{c1}.

$\mathcal{I} = \{86377535B, 26455535C, 94768235H\}$
$\mathcal{P} = \mathcal{I} \cup \{25896335J$

$S \rightarrow R_{p1}|R_{p2}|R_{p3}|R_{p4}$ Símbolo inicial

$R_{p1} \rightarrow Nd\ Nd\ Nd\ Nd\ Nd\ Nd\ R_{c1}\ R_{c2}\ (B|b)$
$R_{p2} \rightarrow Nd\ Nd\ Nd\ Nd\ Nd\ Nd\ R_{c1}\ R_{c2}\ (C|c)$ No terminales ejemplos positivos
$R_{p3} \rightarrow Nd\ Nd\ Nd\ Nd\ Nd\ Nd\ R_{c1}\ R_{c2}\ (H|h)$
$R_{p4} \rightarrow Nd\ Nd\ Nd\ Nd\ Nd\ Nd\ R_{c1}\ R_{c2}\ (J|j)$ Nuevo ejemplo positivo incorporado

$R_{c1} \rightarrow 3$ Reglas de inducción
$R_{c2} \rightarrow 5$

Código 19. Estado de la gramática G'tras aplicar el algoritmo de inducción con el sub-alfabeto Σ_{c1}. Se han creado las reglas R_{c1} y R_{c2}. En el proceso se ha incorporado un nuevo ejemplo positivo propuesto por el algoritmo y validado por el usuario, que pasa a formar parte de la gramática. Los terminales en los cuerpos de producción son transformados a Σ_{c2}, excepto los contenidos en las reglas inducidas.

6.5. Fase II: creación de funciones

En la siguiente fase se analizan los atributos a nivel token con el objetivo de crear reglas, en este caso a modo de funciones, que pueden asociarse a los no terminales en las gramáticas IEG. En este caso las funciones sólo podrán ser asociadas a no terminales cuyo cuerpo de producción sea un token. Para ello, en primer lugar se aplica un proceso de tokenización sobre G' que consiste en agrupar en tokens (cadenas alfanuméricas) los cuerpos de producción de cada R_p, donde cada token vendrá representado con un no terminal R_t (Código 20). En este punto es posible además incorporar tokens en una ventana de contexto para cada ejemplo positivo, cuyo tamaño es parametrizable. Estos tokens de contexto serán incorporados como tokens genéricos a los cuerpos de producción de R_p.

```
S → R_p1|R_p2|R_p3|R_p4 Símbolo inicial

R_p1 → R_t1
R_p2 → R_t2                 No terminales ejemplos positivos
R_p3 → R_t3
R_p4 → R_t4

R_c1 → 3                    Reglas de inducción
R_c2 → 5
R_c3 →\pNd{6}
R_c4 →\pLu

R_t1 → R_c3 R_c1 R_c2 R_c4
R_t2 → R_c3 R_c1 R_c2 R_c4  No terminales derivados de la tokenización:
R_t3 → R_c3 R_c1 R_c2 R_c4  cada no terminal apunta a un token
R_t4 → R_c3 R_c1 R_c2 R_c4
```

Código 20. Gramática G' una vez finalizada la fase de análisis de atributos a nivel carácter, donde nuevas reglas de inducción, R_{c3} y R_{c4} han sido creadas. El proceso de tokenización crea los no terminales R_{t1}-R_{t4}, que agrupan en tokens los cuerpos de producción de R_{p1}-R_{p4}.

De la gramática así configurada se obtendrá una gramática G_i'' para cada atributo $g_i \in \mathcal{G}$ que se quiera incorporar. En ellas se sustituyen los tokens por los valores del atributo aplicado sobre ellos (Código 21). Cada gramática G_i'' tendrá entonces como alfabeto los valores correspondientes al atributo g_i, es decir, $Y_{g_i} \subseteq \Sigma_t$. Sobre cada una de estas gramáticas G_i'' será aplicado el mismo algoritmo de inducción de reglas que en la fase anterior. Las reglas resultantes (esta vez sobre G_i'') serán incorporadas como funciones a los tokens correspondientes de G'. A grandes rasgos el pseudocódigo es el que vemos en el Código 22.

```
S → R_p1|R_p2|R_p3|R_p4 Regla inicial

R_p1 → R_t1
R_p2 → R_t2                 Reglas ejemplos positivos
R_p3 → R_t3
R_p4 → R_t4

R_t1 → 9
R_t2 → 9                    Valor del atributo longitud sobre los tokens
R_t3 → 9
R_t4 → 9
```

Código 21. Gramática G'' procedente de G' al aplicar el atributo longitud sobre los tokens en los cuerpos de producción (excepto en los de las reglas inducidas). El algoritmo de inducción sobre G'' puede generar una regla que comprimiría el valor "9" en R_{t1} - R_{t4}. Esa nueva regla sería aplicada en forma de función en G' a los no terminales de R_{t1} a R_{t4}.

Es importante destacar que mientras que en la fase de atributos a nivel carácter los sub-alfabetos tienen un orden de aplicación claro, comenzando con Σ_{c1} y finalizando con Σ_{c3}, no

ocurre lo mismo en la fase de tokens, donde los sub-alfabetos están formados por los valores de los diferentes atributos sin un orden predefinido. Es el usuario el que debe decidir el orden de aplicación.

```
foreach (Σ_{c_i} in Σ_c) do
         G' ←Transform(G', Σ_{c_i})
         RuleInduction(G')
endforeach

G' ←AddContext(G')
G' ←Tokenize(G')

foreach (g_i in 𝒢) do
        G'' ←Transform(G', Y_{g_i})
         RuleInduction(G'')
        G' ←AddFunctions(G'')
endforeach
```

Código 22. Algoritmo de generación de gramáticas IEG

6.6. Inducción de reglas

El objetivo del algoritmo de inducción es fijar mediante reglas (a modo de nuevos no terminales en la fase I, o de nuevas tuplas (g, y) en la fase II) los patrones localizados en las entidades. Constituye por tanto el núcleo de la metodología, puesto que un exceso en la inducción de reglas llevaría a una gramática muy específica y el consecuente *overfitting* (especialmente en fases tempranas a nivel de caracteres) y lo contrario generaría reglas muy genéricas con el consecuente ruido en la identificación.

6.6.1. Tipos de patrones

Los patrones susceptibles de ser reconocidos son de diversos tipos. Los más habituales en texto son los que denominaremos de *ocurrencia*: la aparición de un mismo símbolo en todos los ejemplos positivos. Por ejemplo, ante un conjunto de correos electrónicos, la arroba podría constituir un patrón de ocurrencia. Pero además de la ocurrencia de ciertos elementos, puede tenerse en cuenta su posición en cada entidad. A este tipo de patrones los llamaremos *posicionales*. Es lo que ocurre por ejemplo con las letras de cierto tipo de

matrículas de coches, que siempre aparecen en la misma posición. Además esta posición puede ser relativa respecto a otros símbolos. Es el caso por ejemplo de la letra del DNI, que aparece siempre tras el último número (o al final de la entidad), independientemente de la longitud de este (ya sean 7 u 8 números).

El algoritmo de inducción de reglas aquí descrito se centrará en el reconocimiento de patrones posicionales absolutos y relativos, detectando también patrones tan sólo de ocurrencia cuando no existan posicionales. Este proceso se aplica tanto a las entidades propiamente como a su contexto, pero de forma independiente. En el primer caso se aplica a los cuerpos de producción de R_p que contienen los terminales de la propia entidad y las reglas inducidas en fases previas en su caso (los terminales pertenecerán a Σ_c o bien a Σ_t, dependiendo de si estamos en la fase I o en la fase II). En el caso del contexto, por el contrario, se aplicará a los terminales y reglas inducidas del contexto izquierdo concatenados a los terminales y reglas inducidas del contexto derecho de la entidad. De este modo se consigue evitar ruido en la generación de patrones mezclando la entidad con el contexto, al tiempo que se flexibiliza la identificación de potenciales patrones contextuales que no necesariamente aparecen siempre a izquierda o derecha de la entidad.

Gran parte de los sistemas de aprendizaje simbólico de tipo cobertura, como Whisk y Amilcare, generan para cada entidad conocida todas las posibles combinaciones de los valores de los atributos considerados sobre sus tokens, con mayor o menor grado de generalización (e.g. el token "at" puede representarse, de menor a mayor grado de generalización, con el texto "at", como una preposición o como un token cualquiera). Seleccionan entonces la mejor combinación en función de los resultados estimados sobre el corpus, combinadas en algunos casos con medidas de complejidad de la propia regla. Esto supone contar con un número suficiente de anotaciones conocidas en el corpus. Un problema adicional es tratar de reducir el espacio de búsqueda, y la aplicación de esta técnica a atributos a nivel carácter generaría un espacio de búsqueda intratable.

Por otro lado, los sistemas de tipo compresión sí analizan conjuntos de entidades para detectar patrones. Es el caso de RAPIER, que sí analiza la existencia de patrones posicionales para generar las reglas, pero sólo compara las entidades a pares, y es muy

restrictivo en cuanto a la alineación de los símbolos en común (sólo considera posiciones absolutas). Esto último también ocurre en el trabajo de Brauer, donde estos patrones se localizan únicamente al comienzo o fin de cada entidad, como prefijos y sufijos. A diferencia de estos casos, el algoritmo de inducción descrito en esta sección ofrece la flexibilidad suficiente para analizar todos los ejemplos a la vez en busca de patrones posicionales sin exigir alineación exacta de los símbolos. Dado que debemos asumir además que no disponemos de gran cantidad de información anotada para hacer estimaciones, se utilizarán las técnicas de aprendizaje activo para la validación, como se verá más adelante en este capítulo.

6.6.2. Identificación de patrones

Los patrones posicionales potenciales están formados por las posiciones de un mismo símbolo terminal en cada una de las entidades (o contexto) representados en R_p (Tabla 24). Los símbolos terminales sobre los que se crean han de ser por tanto comunes a todas las entidades.

Podemos definir un potencial patrón posicional o *path* como el resultado de una función p tal que:

$$p\colon \left\{t \in \{\Sigma_c \cup \Sigma_t \cup V\} \middle| \forall R_{pi} : R_{pi} \rightarrow \alpha t \beta\right\} \rightarrow \mathbb{N}^{|\mathcal{P}|}$$

$$\alpha, \beta \in \{\Sigma_c \cup \Sigma_t \cup V\}^* \cup \{\varepsilon\}$$

donde el dominio t es un símbolo de la gramática que puede ser tanto terminal como no terminal y aparece en todos los cuerpos de producción de R_p, y el rango es un vector donde cada dimensión i corresponderá a la posición del símbolo t en R_{pi}.

A estos *path* se añaden dos:

- *Path* inicial p_o: $\forall R_i : p_{oi} = 0$
- *Path* final p_f: $\forall R_{pi} : p_{fi} = length(símbolos\ en\ R_{pi})$

El objetivo será entonces identificar el *path* de terminales mejor alineados con respecto a otros *path* cualesquiera. La idea detrás es que cuanto mejor sea el alineamiento, más

probable es que se trate de un patrón posicional. Por ejemplo, si existiera un patrón posicional con posiciones fijas (no dependientes de otros elementos) su *path* aparecerá perfectamente alineado respecto al *path* inicial. Si por el contrario depende de otros, aparecerá alineado con respecto a ellos, ya sean otros terminales o no terminales inducidos en fases previas. El *path* de un terminal con mejor grado de alineamiento respecto a otros *path* cualesquiera será finalmente convertido en regla de producción. Esta conversión consiste en transformar en cada R_{pi} el terminal en la posición indicada en el *path* por una regla de inducción. Esta regla tendrá como cuerpo de producción al terminal sustituido.

Tabla 24. Patrones posicionales potenciales en una gramática con las entidades $R_{p1} = 972 - 830165$ y $R_{p2} = 972 - 563410$

Terminales comunes	**Posibles patrones: {posición terminal en R_{p1}, posición terminal en R_{p2}}**
9	{0,0}
7	{1,1}
2	{2,2}
-	{3,3}
8	-
3	{5,6}
0	{6,9}
1	{7,8}
6	{8,5}
5	{9,4}
4	-

El algoritmo presenta entonces dos problemas a resolver:

- Medir el grado de alineamiento entre *paths.*
- Identificar los *paths* en el caso de terminales repetidos.

Grado de alineamiento

El grado de alineamiento l se mide por cada par de *path* y responde a la siguiente fórmula:

$$l(p_a, p_b) = \begin{cases} 0 & cross(p_a, p_b) = 1 \\ 1 - \sum_{i=1}^{|\mathcal{P}|-1} \frac{|(p_{b(i+1)} - p_{a(i+1)}) - (p_{bi} - p_{ai})|}{|R_p|} & otherwise \end{cases}$$

donde la función *cross* determina si dos *paths* se cruzan o solapan en algún punto. Ante esta circunstancia el grado de alineamiento entre ambos *paths* pasa a ser nulo. Por el contrario, si p_ay p_b resultan alinearse perfectamente, es decir, sus posiciones relativas son iguales para cada entidad, entonces su valor será 1. El objetivo entonces será identificar el valor máximo de alineamiento y los *paths* de terminales que tengan ese grado de alineamiento con respecto a cualquier otro. Esos terminales serán entonces susceptibles de convertirse en patrones verticales.

Identificación de *paths* ante terminales repetidos

Como hemos visto, se calculan *paths* para cada símbolo que aparezca en todos los R_p de la gramática. Pero en algunos casos un mismo terminal puede aparecer más de una vez en cada R_p, como ocurre con los terminales "2", "-", "4", "3" y "0" en la Tabla 25.

Tabla 25. Patrones posicionales potenciales en una gramática con las entidades $R_{p1} = 972 - 243 - 0120$ y $R_{p2} = 972484 - 9330$

Terminales comunes	**Posibles patrones: {posición terminal en R_{p1}, posición terminal en R_{p2}}**
9	{0,0}
7	{1,1}
2	{2,2}{4,2}{10,2}
-	{3,6}{7,6}
4	{5,3}{5,5}
3	{6,8}{6,9}
0	{8,10}{11,10}

Cuando la ocurrencia múltiple se da entre terminales adyacentes, estos son agrupados en un único símbolo (compresión). Para la creación de *paths* estos símbolos serán contabilizados como uno solo con la posición del primero de ellos, pero su cardinalidad es almacenada porque será utilizada si finalmente se forma regla. El objetivo es, además de la eficiencia en el cálculo, la posibilidad de crear reglas de inducción que a su vez compriman la gramática. Si un *path* de este tipo da lugar finalmente a una regla, entonces su cuerpo de producción será el terminal en cuestión con la cardinalidad mínima de todas las dimensiones de ese *path*. Así, si por ejemplo tenemos números de cuentas corrientes como

entidades con 20 números, al convertirlos a Unicode (Σ_{c2}) se creará un único *path* alineado perfectamente a p_o, y por tanto una potencial regla de inducción del tipo $R_c \rightarrow \backslash pNd\{20\}$ que sustituiría los cuerpos de producción de todas las regla R_p. Pero si esas entidades tuvieran entre 18 y 20 dígitos por ejemplo, entonces la regla, de crearse, sería $R_c \rightarrow \backslash pNd\{18\}$, sustituyendo a 18 dígitos únicamente y por tanto dejando en los cuerpos de producción de R_p los sobrantes.

Tabla 26. Aplicación de la heurística al caso de un terminal "a" que aparece en diferentes posiciones en cada R_{pi}. Para cada posición alternativa de "a" en los diferentes R_{pi} (posiciones 0, 5, 6 y 7) se crea un *path* con las posiciones de "a" más cercanas. Estos *path* son: (0,0,0,0,0,0), (5,5,6,7,7,6) y (7,7,6,7,7,6).

	Pos. 0	**Pos. 1**	**Pos. 2**	**Pos. 3**	**Pos. 4**	**Pos. 5**	**Pos. 6**	**Pos. 7**
R_{p1}	a					a		a
R_{p2}	a					a		a
R_{p3}	a						a	
R_{p4}	a							a
R_{p5}	a							a
R_{p6}	a						a	

Cuando los terminales repetidos no son adyacentes, el número de los *path* resultantes puede aumentar considerablemente, ya que por cada uno de estos terminales tendríamos que el número total de *paths* es $n_1 \cdot n_2 \cdot n_3 \cdot \ldots \cdot n_{|\mathcal{P}|}$ siendo n_i el número de posiciones diferentes (no adyacentes) en R_{pi}. Suponiendo que n es el máximo de posiciones alternativas en cualquier R_{p_i}, el número de *paths* es entonces $n^{|\mathcal{P}|}$. La complejidad es por tanto exponencial con respecto al número de entidades $|\mathcal{P}|$, por lo que es necesaria una heurística que nos permita seleccionar los mejores *path* de forma eficiente.

La heurística aplicada prioriza la identificación de potenciales patrones posicionales absolutos. El algoritmo es el siguiente. Para cada terminal con más de una aparición en al menos una entidad, se identificarán las diferentes posiciones en el conjunto de entidades. Para cada posición diferente, se seleccionarán los terminales más cercanos de cada entidad por la derecha y por la izquierda (ver Código 23). Las posiciones de los terminales seleccionados darán como resultado los *paths* mejor alineados a p_0. El pseudocódigo puede verse en el Código 23.

De este modo, en el peor de los casos por cada terminal tendríamos una complejidad de $O(|\mathcal{P}| \cdot n^2)$, pasando de un orden exponencial a lineal respecto al número de entidades $\mathcal{P}$, y de polinomial a cuadrático con respecto a la longitud de las mismas.

```
let terminalList be the terminals that appear in all entities and more
than once in at least one entity.
let HeuristicPaths be an empty array of paths

foreach terminal in terminalList
    let listPositions be an empty array of integer.

    foreach R_pi in R_p
            listPositions ←listPositions ∪ Positions(terminal, R_pi)
    listPositions ←Unique(listPositions)

    foreach position in listPositions
         foreach R_pi in R_p
             foreach pos in (Positions(terminal, R_pi)<= position)
                   pIzq_i ← Positions(Min(|position-pos|))
             endforeach
             foreach pos in (Positions(terminal, R_pi)<= position)
                   pDcha_i ← Positions(Min(|position-pos|))
             endforeach
         endforeach
     endforeach

     HeuristicPaths = HeuristicPaths ∪ Unique(pIzq ∪ pDcha)
endforeach
```

Código 23. Heurística para la generación de *paths*.

6.6.3. Clustering

El algoritmo de inducción planteado aporta una enorme flexibilidad en la identificación de patrones puesto que no exige un alineamiento exacto de los terminales para que puedan ser creados, lo que es muy útil en entidades de distinto tamaño. Por el contrario, exige que el terminal sobre el que puedan crearse aparezca en todas las entidades. Esta condición es muy restrictiva e implicaría que todos los ejemplos siguen exactamente un mismo patrón. En la realidad nos encontramos con que esto no siempre es así. Como ejemplo imaginemos que queremos capturar matrículas de vehículos, con entidades que corresponden a matrículas antiguas mezcladas con las actuales. Si esas matrículas son españolas, podríamos tener entonces mezclados hasta tres patrones diferentes (suponiendo matrículas no especiales):

- Matrículas españolas de 1900 a 1971: $Lu\{1,3\}\ Nd\{1,6\}$
- Matrículas españolas de 1971 a 2000: $L\{1,2\}\ Nd\{4\}\ Lu\{1,2\}$
- Matrículas españolas a partir del 2000: $Nd\{4\}\ Lu\{3\}$

Si tratáramos de buscar patrones en todas estas entidades a la vez obtendríamos, en el mejor de lo casos, un patrón similar a $Lu\{0,3\}\ Nd\{1,6\}\ Lu\{0,3\}$. Este patrón, aunque es muy concreto, no representa exactamente las entidades a capturar, puesto que reconocería como válida por ejemplo una matrícula formada únicamente por números, lo que probablemente generaría mucho ruido en corpus con otros tipos de entidades numéricas, como por ejemplo códigos postales. Otro factor a considerar son los *outliers* que pudiera haber entre las entidades o incluso errores en los ejemplos proporcionados, que podrían llevarnos a perder información muy valiosa para la creación de patrones.

Por estos motivos se realiza una fase previa a la aplicación del algoritmo de inducción de reglas explicado anteriormente, en la que se agrupan por similitud las entidades. Será a estos grupos a los que se les aplique de modo independiente el algoritmo de inducción de reglas, con lo que será posible encontrar en nuestra gramática G' reglas de inducción que no existen en todos los cuerpos de producción de R_p, generando así patrones diferentes para un mismo conjunto de ejemplos.

Para ello se ha implementado un algoritmo de clustering jerárquico aglomerativo con máxima distancia entre elementos (complete linkage clustering) [Orallo et al., 2005]. Se utiliza Levenshtein como medida de distancia [Levenshtein, 1966], con un valor de 1 para borrados, sustituciones y adiciones. Esta distancia de edición es aplicada sobre cada par de entidades y normalizada por la longitud de la secuencia de terminales más larga. Una vez obtenidos estos clusters, donde cada uno de ellos corresponde a una representación parcial de G', se aplica el proceso de inducción de reglas sobre cada uno de ellos y el resultado es actualizado en G'. El pseudocódigo completo del algoritmo, incorporando ahora el proceso de clustering, puede verse en el Código 24.

```
foreach (Σ_ci in Σ_c) do
        G' ←Transform(G', Σ_ci)
        Grammars ←Clustering(G')
        foreach grammar in Grammars
            (G',P,N) ←RuleInduction(grammar)
        G' ←Update(G', grammar)
        endforeach
endforeach

G' ←AddContext(G')
G' ←Tokenize(G')

foreach (g_i in G) do
        G'' ←Transform(G', Y_gi)
        Grammars ←Clustering(G'')
        foreach grammar in Grammars
            RuleInduction(grammar)
            G'' ←Update(G'', grammar)
            G' ←AddFunctions(G'')
        endforeach
endforeach
```

Código 24. Algoritmo de generación de gramáticas IEG con el proceso de clustering incorporado. Los cambios introducidos respecto al algoritmo del Código 22 aparecen en un recuadro.

6.6.4. Aprendizaje activo

Un aspecto importante en cualquier algoritmo de generación de patrones es el modo de validar los resultados. Esta validación es la que realmente determina cuáles de las potenciales reglas serán finalmente creadas. Cuando se cuenta con un corpus anotado, son estos ejemplos, tanto positivos como negativos, los que se usan para ello. En nuestro caso se utilizarán técnicas de aprendizaje activo capaces de aportar ejemplos que sirvan no sólo para hacer patrones más exhaustivos, sino también para validar los propios patrones que van siendo creados.

Para ello, en cada iteración del algoritmo, donde un conjunto de potenciales reglas de inducción son generadas, se seleccionan una serie de ejemplos del corpus no anotado para que el usuario los valide. Los ejemplos positivos validados por el usuario son incorporados a la gramática, y se repetirá el proceso de generación de potenciales reglas de inducción. Si las nuevas reglas generadas coinciden con las potenciales reglas anteriores, entonces se crearán como reglas de inducción en la gramática y se continuará el proceso hasta que no puedan ser creadas nuevas reglas (Código 25).

El factor clave en este proceso será el modo de seleccionar los ejemplos a validar. El objetivo que se persigue es doble:

- Localizar *contraejemplos* que pudieran impedir la creación de reglas propuestas, supliendo así la carencia de ejemplos negativos y posibilitando una mejora en precisión.
- Localizar *nuevos ejemplos* que amplíen el abanico de ejemplos positivos, supliendo así la carencia de ejemplos positivos y posibilitando una mejora en exhaustividad.

  Para los contraejemplos se seleccionarán aquellos fragmentos de texto que encajan en todo salvo en las nuevas reglas propuestas (al menos una), y se mostrarán ordenados de mayor a menor número de reglas propuestas con las que sí encaja. La selección de nuevos ejemplos es menos restrictiva, y simplemente se ordenan los fragmentos de texto por número de terminales con los que encajan, valorándose aquellos que pertenecen a la entidad el doble que los que pertenecen al contexto. El usuario deberá validar tanto los contraejemplos como los nuevos ejemplos, y aquellos considerados entidades válidas serán incorporados a la gramática. Se adaptarán entonces a los patrones ya creados de otra entidad ya existente o en caso contrario se considerarán *outliers* respecto a los patrones ya creados y incorporarán de la forma más específica posible para continuar con el proceso.

```
let PreviousRules be an empty array of induction rules
let Grammar be the formal grammar where we have to find potential
rules
NewRules = GetPotentialRules(Grammar)
while (NewRules is not empty)
      if (NewRules ⊆ PreviousRules)
         Add(NewRules, Grammar)
     else
         Examples ←ProposeNewEntities(Grammar, NewRules)
         foreach example in Examples do
              if (IsEntity(example))
                  𝒫 ←𝒫 + example
                  Add(example, Grammar)
              else
                  𝒩 ←𝒩 + example
              endif
         PreviousRules ←NewRules
     endif
     NewRules ←GetPotentialRules(Grammar)
endwhile
```

Código 25. Algoritmo de inducción de reglas (RuleInduction) con la incorporación de aprendizaje activo.

En cuanto a la condición de parada, a diferencia de otros algoritmos de aprendizaje activo, no se requiere una estimación de efectividad. Las condiciones de parada suelen venir dadas por un mínimo o máximo en la efectividad estimada del algoritmo o bien por convergencia, cuando más iteraciones no varían los resultados [Laws & Schütze, 2008]. La condición de parada en nuestro caso viene dada por el número de alfabetos y la identificación de potenciales patrones posicionales en cada alfabeto y fase. Estos factores dan a la metodología un criterio de parada interno (no parametrizable) y dependiente además de la entidad analizada y los ejemplos proporcionados.

6.7. Conclusiones

Se ha diseñado una metodología de generación de patrones para el reconocimiento de entidades evitando el uso de corpus anotados, sin renunciar por ello a la potencia y flexibilidad suficiente para adaptarse a nuevos tipos de entidad y dominios.

Se trata de un método ad-hoc con técnicas de aprendizaje activo. El método parte de un conjunto de ejemplos de entidades y un corpus no anotado, y sigue un proceso bottom-up de compresión. Parte de las propias entidades como potenciales reglas de la gramática IEG y las generaliza progresivamente, comprimiendo con ello la gramática. La combinación de un algoritmo capaz de identificar patrones de ocurrencia y posicionales con técnicas de clustering permite la generación de patrones cuando se dispone de escasa información anotada. A esto contribuye la tolerancia del algoritmo de inducción de reglas posicionales, que a diferencia de otras técnicas, no requiere un alineamiento perfecto de los símbolos en las diferentes entidades para su detección. Las técnicas de aprendizaje activo hacen uso del conocimiento del usuario de un modo dinámico para validar estas reglas y ampliar el corpus de entrenamiento con nuevos ejemplos, seleccionando los fragmentos a etiquetar más útiles en cada momento.

La metodología ha sido diseñada para presentar al usuario directamente la secuencia de tokens que considera que pueden ser entidad, en lugar de pedirle la anotación de

documentos completos o secuencias prefijadas como ocurre en otras metodologías. El objetivo es evitar dependencias de los tipos de entidad y de las características documentales del corpus.

Como ocurre en los clasificadores, y a diferencia en general de los métodos ad-hoc y de bootstrapping, esta metodología es lo suficientemente flexible para incorporar cualquier atributo sin modificar el código, con el límite de que han de ser atributos a nivel token. Incorpora además atributos a nivel carácter de forma automática. La combinación de ambos tipos de atributo genera gramáticas de tipo IEG que, como vimos en el Capítulo 5, son modelos comprensibles, editables y con cierto grado de estandarización. En cualquier caso, a pesar de las ventajas ofrecidas por el IEG, el método de generación puede utilizarse también para la generación de otros modelos de representación de patrones. Los atributos a nivel carácter y token pueden representarse en dos autómatas distintos mediante autómatas finitos en cascada, mientras que el uso tan sólo de atributos a nivel carácter (fase I) lleva a la generación de expresiones regulares estándar. En todos los casos la potencia expresiva es la de las expresiones regulares.

La flexibilidad de la metodología aquí descrita ante la incorporación de diferentes atributos y la potencia expresiva de los patrones que genera permite aplicarla a textos y entidades de muy diversas características. El único requisito es contar con un corpus no anotado y un pequeño número de semillas iniciales, que pueden o no estar contenidas en él. Es posible incluso la aplicación a diferentes idiomas con el alfabeto latino, aunque las especificidades de cada idioma harían aconsejable futuros estudio en lo que se refiere a aspectos de tokenización del texto y agrupación de clases de caracteres similares.

Los resultados obtenidos en cuanto a eficacia y costes de anotación son objeto del siguiente capítulo.

Capítulo 7
Evaluación y resultados

If you can't measure it, you can't improve it -Lord Kelvin

7.1. Introducción

El método de generación de gramáticas IEG ha sido evaluado haciendo uso de dos corpus utilizados frecuentemente en el ámbito de la Extracción de Información: el *Seminar Announcement* corpus y el *Software Jobs* corpus[24].

El primero, *the Seminar Announcement Corpus* (seminarios), es una colección de 485 anuncios de seminarios recopilados por Dayne Freitag. Cada uno de los documentos consiste en un anuncio de un seminario, con el tiempo de inicio (*startTime)*, ponente (*speaker)*, localización (*location)* y, en algunos casos, tiempo de fin (*endTime*).

El *Software Jobs Corpus* (jobs) consiste en 300 documentos, cada uno con una oferta de trabajo en el ámbito de la informática. Fue recopilado por Mary Elaine Califf y anotados 17 tipos de entidad: *ID*, *Title*, *Salary*, *Company*, *Recruiter*, *State*, *City* , *Country*, *Language* (e.g., VISUAL BASIC), *Platform*, *Application* (e.g., SQL Server), *Area, Req. years experience, Desired years experience, Req. degree, Desired degree,* y *Post date*. Adicionalmente hemos añadido a este

[24] Accesibles en: http://www.isi.edu/info-agents/RISE/repository.html

corpus una nueva entidad: *phone*, que será incluida en las evaluaciones del mismo modo que el resto. Para ello se ha elaborado manualmente una expresión regular específica de este corpus (Código 26) con la que ha sido posible capturar 651 números de teléfono, sin falsos positivos y con unas tasas de recall elevadas (lo habitual es encontrar dos números de teléfono -teléfono y fax- por cada uno de los 300 documentos).

```
((?<=\W|^)\(?                          Símbolos no alfanuméricos previos
\d{3}[\)\\\-\/]?\s*                    Código de Area de 3 dígitos
(\d{3}\W(\d{4}|USSI))(\s*x\d\d\d)?(?=\W|$))   Teléfono
|                                      ó
((?<=fax#\s*)\d{3}\-\d{4})             Fax
```

Código 26. Expresión regular para la captura de números de teléfono (entidad *phone*) en el corpus jobs.

A ambos corpus se les ha aplicado el procesador de textos *open-source* ANNIE, que es parte de la herramienta GATE [Cunningham et al., 2013], para la obtención de los valores de los atributos a nivel token que se usarán. De los atributos lingüísticos que esta herramienta es capaz de extraer se ha seleccionado la información sobre mayúsculas (*lowercase, upperInitial, allCaps, mixedCaps*), la categoría morfo-sintáctica (*Part of Speech* o POS), el tipo de token (*word, number* o *punctuation*) y su *stem*, además del propio token. Aunque ANNIE también realiza análisis semántico y este tipo de atributos son habitualmente utilizados en las herramientas de extracción de información (bien mediante listados, recursos como WordNet, expresiones regulares o a través del uso de otras herramientas de anotación), sólo se ha utilizado la información lingüística básica. De este modo se evitan grandes variaciones dependiendo de los recursos concretos que se utilizan y las frecuentes actualizaciones que sobre ellos se realizan.

El objetivo de nuestra metodología es la generación de gramáticas IEG para cada uno de los tipos de entidad de forma independiente (el reconocimiento de un tipo de entidad no requiere el reconocimiento de otros tipos de entidades, lo que sí suele ocurrir en otros patrones generados habitualmente con aprendizaje automático). Se evaluará por tanto la capacidad de la misma, en términos de efectividad, para adaptarse a 22 tipos de entidades diferentes en dos corpus distintos, así como el coste de anotación requerido. Estas evaluaciones se realizarán sobre dos tareas principales:

- Tarea A: capacidad de la metodología para generar patrones capaces de capturar tipos de entidad concretos, ya sean gramáticas IEG o expresiones regulares.
- Tarea B: capacidad de la metodología para ayudar en la creación de recursos que permitan la inclusión de nuevos tipos de entidad en los foros, ya sea mediante la anotación de un corpus o en la adquisición de semillas que puedan servir de entrada a otras herramientas.

Se contará únicamente con un conjunto de semillas iniciales y un corpus no anotado (bien de seminarios, bien de jobs, dependiendo de la entidad). Para la tarea A es necesario disponer de un corpus de test diferente del corpus de entrenamiento. Es habitual en aprendizaje activo utilizar todo el corpus para entrenar y evaluar la eficacia de los patrones generados con aquello (generalmente documentos) no anotados durante el proceso. Esto supone una clara ventaja para estos algoritmos, ya que pueden recopilar información sobre el corpus de test. Nosotros sin embargo evitaremos que el corpus sobre el que se valida sea en ningún momento visto por el algoritmo, que es la aproximación utilizada habitualmente en los métodos de aprendizaje supervisado. Para ello se evaluarán los resultados con un tercio del corpus (corpus de test), mientras que los restantes dos tercios (corpus de entrenamiento) se usará como corpus de entrada no anotado de donde tomar los ejemplos a validar por el usuario.

Para la tarea B se utilizará el corpus completo para entrenar y la validación se realizará sobre todo el corpus a excepción de las semillas iniciales utilizadas.

En ambos casos las semillas serán tomadas aleatoriamente del corpus (en el primer caso, del corpus de entrenamiento). La efectividad será medida en términos de precision, recall y la combinación de ambas con macro-averaged F tomando $\beta = 1$. Los costes para el usuario se medirán en términos de ejemplos validados y número total de tokens que los forman y que por tanto ha debido revisar (las semillas iniciales serán también contabilizadas como costes de anotación).

7.2. Configuración

La metodología presentada en el Capítulo 6 permite la configuración de ciertos parámetros en las herramientas donde sea implementada. En concreto son configurables los siguientes valores:

- Máximo ejemplos mostrados para validar: número máximo de *contraejemplos* (ejemplos que pudieran impedir la creación de reglas propuestas) o *nuevos ejemplos* (que amplíen el abanico de ejemplos positivos) a mostrar durante el proceso de aprendizaje activo. Se mostrarán 10 ejemplos de cada tipo en cada iteración.
- Mínimo ejemplos mostrados por regla: número mínimo de *contraejemplos* para cada regla candidata. Tiene prioridad frente al parámetro anterior. Se considerará que el mínimo son 2 ejemplos por regla candidata.
- Unidad de anotación: fragmento de texto que se usará para validar. Se mostrará al usuario la potencial entidad con una ventana de 10 caracteres en cada extremo. Se considerará válida una entidad propuesta si en su ventana de más/menos 10 caracteres se solapa con una válida. Esto permite cierta flexibilidad ante errores en la detección de los límites durante el proceso de aprendizaje sin aumentar significativamente la carga del usuario, puesto que el tamaño de la ventana de texto ofrecida al usuario es mínima. En cualquier caso, para medir los costes de anotación serán computados todos los tokens que el usuario habría de revisar.
- Contexto: número de tokens en cada extremo de la entidad que se utilizará como contexto en la fase II (en la fase I siempre se toma como contexto el carácter inmediatamente anterior y el inmediatamente posterior).
- Umbral de similitud: umbral de similitud entre entidades para agruparlas en un mismo *cluster*. Esto condicionará la formación de reglas entre entidades con mayor o menor similitud en la propia entidad y su contexto. Su valor está comprendido entre 0 (entidades exactamente iguales) y 1. Cuanto menor sea el umbral, mayor similitud entre ejemplos se exigirá para la formación de reglas.

Los dos últimos parámetros no han sido prefijados porque dependen en mayor medida del corpus y las entidades a reconocer. Por este motivo, se ha realizado un diseño experimental *full-factorial* con el objetivo de determinar los valores más adecuados. Se ha probado con seis valores de contexto: 1, 2, 3, 4, 6 y 8, y tres valores de umbral de similitud: 0.6, 0.8 y 0.9[25]. Se usa el corpus completo para eliminar posibles sesgos por las diferentes divisiones que pueden hacerse entre corpus de entrenamiento y de test, y se toman 10 semillas aleatorias del corpus. Se han realizado 10 ejecuciones por cada configuración, cada una de ellas con un conjunto diferente y aleatorio de semillas. En total han resultado 3240 ejecuciones sobre jobs y 720 sobre seminarios.

7.2.1. Contexto

La influencia del contexto en la F obtenida es clara, observándose en general un aumento de la misma al aumentar el tamaño del contexto (Figura 17). En ambos casos este crecimiento resulta ser significativo, con una pendiente β =0.013±0.007 en el caso de jobs y β =0.086±0.004 en el caso de seminarios, por lo que desde el punto de vista de la efectividad el contexto más apropiado de entre los estudiados es 8.

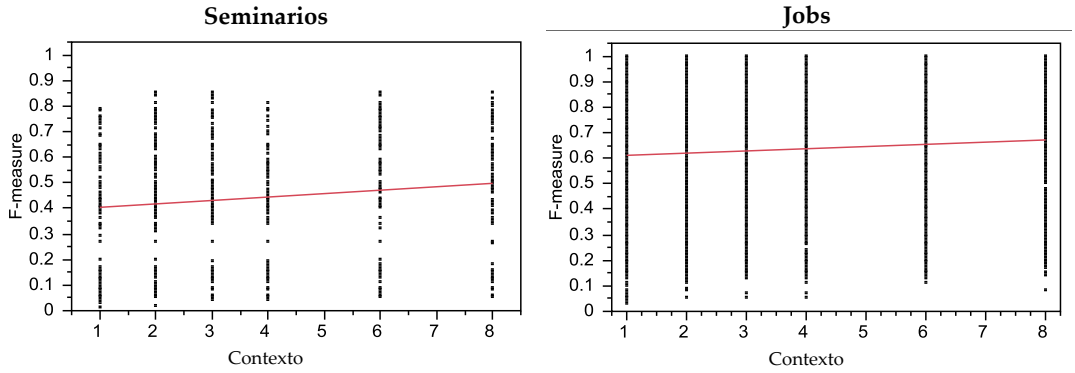


Figura 17. Rectas de regresión para la F-measure dependiendo del contexto para el corpus seminarios y jobs.

Sin embargo también se observa que a mayor contexto, mayor es la cantidad de tokens que es necesario validar, aumentando la carga para el usuario (Figura 18). En concreto el crecimiento en tokens validados también resulta ser significativo en ambos casos, con

[25] Estos valores son los mejores según observaciones previas en experimentos realizados con menor cantidad de ejecuciones.

pendiente β =192.7±43.2 en el caso de seminarios, y β =55.9±12.4 para jobs. Este hecho se debe a que, a mayor contexto, mayor es la cantidad de reglas potenciales, lo que suele traducirse en un incremento en el número de validaciones que se le pide al usuario.

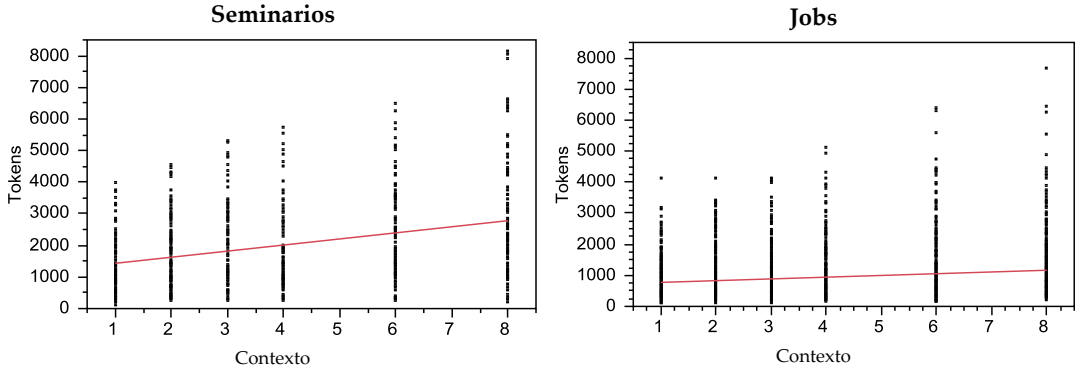


Figura 18. Rectas de regresión para el número de tokens validados dependiendo del contexto para el corpus seminarios y jobs.

Ante estos resultados, el objetivo será seleccionar un contexto que equilibre valores de efectividad con costes de anotación. Para ello se realiza un análisis pormenorizado mediante el test de Tukey-Kramer [Montgomery, 2009] (Sall, Lehman, Stephens, & Creighton, 2012), comparando las diferentes distribuciones de F-measure entre sí según los distintos valores de contexto.

Tabla 27. Agrupación de las distribuciones de F-measure para diferentes valores de contexto según el test de Tukey-Kramer. Las distribuciones en el grupo A son las que obtienen valores más altos para F-measure, mientras que en las de B ocurre lo contrario. Los grupos A y B son significativamente diferentes. El grupo AB no es significativamente diferente de A y de B.

Contexto	Grupo (Seminarios)	Grupo (Jobs)
8	A	A
6	AB	A
4	B	AB
3	AB	AB
2	AB	AB
1	B	B

En la Tabla 27 puede verse que resultan dos grupos: A y B. Para niveles de contexto que resultan en el mismo grupo no se observan diferencias significativas, mientras que niveles que aparecen en ambos grupos indican diferencias significativas con algunos de los niveles que contiene pero no con todos. Por ejemplo, hay una diferencia significativa entre contextos 8 (A) y 1 (B), pero no entre contextos 3 (AB) y 1, o contextos 2 y 8. Teniendo en cuenta estos datos, tomaremos el valor 2 como contexto, puesto que resulta ser el valor más

bajo de contexto (y por tanto con menos coste de anotación) cuya distribución de F-measure no es significativamente diferente de la mejor, esto es, de la obtenida con el valor de contexto 8.

7.2.2. Umbral de similitud

Los resultados obtenidos para el umbral de similitud son variables dependiendo del corpus. Mientras que para el corpus de seminarios el mejor umbral parece ser 0.8, para jobs parece que es mejor cuanto menor valor tenga, es decir, 0.6 (Figura 19).

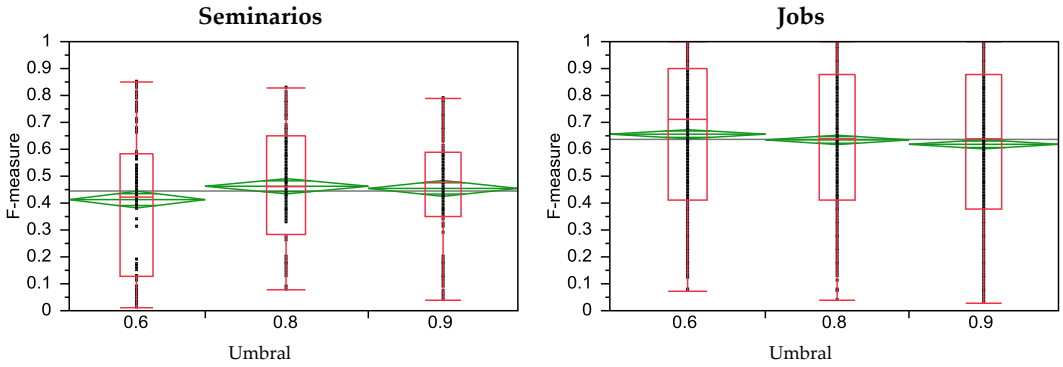


Figura 19. Distribución de la F-measure para cada valor de umbral en el corpus de seminarios y jobs. Los rombos muestran los intervalos de confianza al 95% alrededor de la media.

Un Análisis de Varianza (ANOVA) [Montgomery, 2009][Sall et al., 2012] en el corpus de jobs muestra que las medias de la F-measure entre los distintos valores de umbral son significativamente distintas ($F(2)=4.803$, $p=0.008$). Comparando cada par de umbrales, se observa una diferencia significativa entre los umbrales 0.6 y 0.9 de 0.038 ± 0.024 ($t(2877)=3.088$, $p=0.002$), mientras que el umbral 0.8 no es significativamente distinto al resto.

En el corpus de seminarios no se observan diferencias significativas ($F(2)=2.998$, $p=0.051$) aunque el p-valor roza el límite. De hecho, comparando cada par de umbrales sí observamos una diferencia significativa entre 0.8 y 0.6 de 0.05 ± 0.042 ($t(657)=2.32$, $p=0.021$), mientras que el umbral 0.9 no es significativamente distinto al resto.

Ante estos datos se selecciona el valor 0.8 como valor de umbral para evaluar la metodología con estos corpus. Este valor es con el que resulta mayor tasa de F-measure en

seminarios y aunque no ocurre lo mismo en jobs, no resulta ser significativamente diferente del mejor, que en este caso era 0.6.

En cualquier caso hay que tener en cuenta que la variabilidad observada se debe a que el tipo de entidad influye en la elección del umbral más adecuado, de modo que para unos tipos de entidad resulta positivo aumentarlo mientras que para otros ocurre todo lo contrario o no influye prácticamente nada en la F-measure (Figura 20). Por ejemplo, en el corpus de seminarios un aumento en el umbral influye positivamente en la F-measure de los tipos de entidad *stime* y *speaker*, mientras que ocurre lo contrario con *etime* y *location.* En el caso de jobs, el incremento del umbral afecta positivamente en especial a *company*, *recruiter*, *req_years_experience*, *state* y *language*, mientras que *desired_years_experience*, *id*, *req_degree* y *salary* se ven especialmente afectadas de forma negativa.

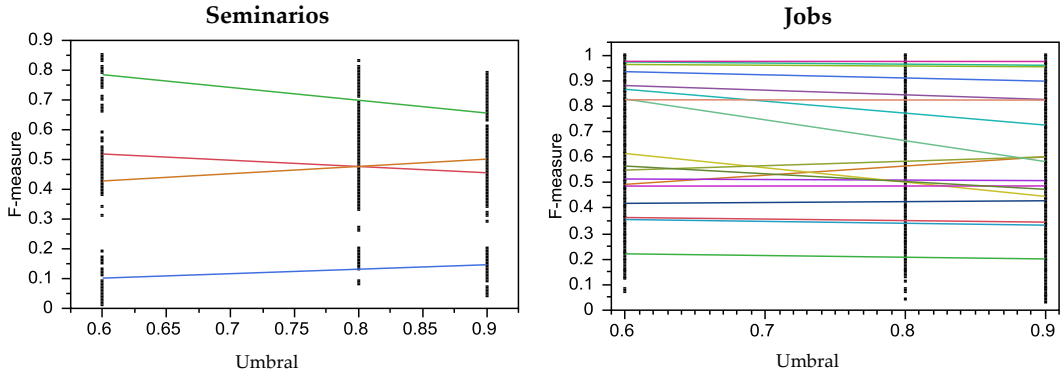


Figura 20. Rectas de regresión, a modo meramente ilustrativo, para la F-measure dependiendo del umbral por cada tipo de entidad del corpus seminarios y jobs.

Otro aspecto a considerar en la elección del umbral es el coste para el usuario. Se ha observado que a menor valor de umbral, más cantidad de tokens necesita validar el usuario (Figura 21). En ambos corpus se observa esta tendencia de forma significativa. Esto se debe a que a menor umbral se generarán más *clusters* con entidades similares, con las consecuentes validaciones por parte del usuario para las reglas potenciales en cada uno de ellos.

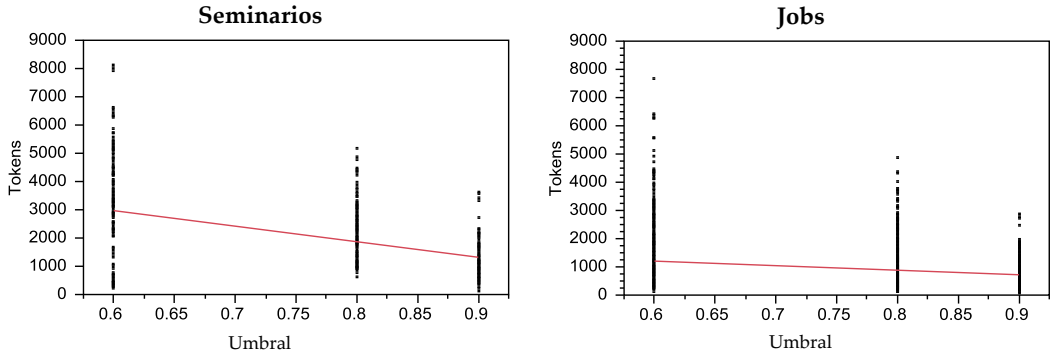


Figura 21. Rectas de regresión para el número de tokens validados dependiendo del umbral para el corpus seminarios y jobs.

Estos factores podrían ser tenidos en cuenta a la hora de ajustar la metodología a la captura de entidades específicas, en lugar de aplicarla a entidades tan diferentes como las contenidas en los corpus utilizados. De hecho, en general las variaciones entre diferentes configuraciones tienen poco impacto en los resultados globales por considerar tipos muy diferentes de entidades a la vez (esto se aprecia especialmente en el corpus jobs).

7.3. Tarea A. Generación de gramáticas IEG

La evaluación ha sido realizada sobre los corpus de seminarios y jobs, a partir de los datos obtenidos para cada uno de los tipos de entidad de forma independiente. Se reportan los resultados medios tanto de coste como de efectividad para el total de las entidades en cada corpus. Se ha probado con la elección de 2, 5, 10, 20, 50 y 100 semillas iniciales, y para cada número de semillas se han realizado diez ejecuciones distintas variando la selección aleatoria de las semillas. Adicionalmente, se han realizado cinco divisiones diferentes del corpus, también aleatorias, en corpus de entrenamiento y corpus de test. El número total de ejecuciones para cada colección ha sido:

Jobs: 18 $tipos\ de\ NE\ \times 6\ nº\ semillas\ \times 5\ nº\ corpus\ \times 10\ selección\ seillas = 5400$

Seminarios: 4 $tipos\ de\ NE\ \times 6\ nº\ semillas\ \times 5\ nº\ corpus\ \times 10\ selección\ semillas = 1200$

resultando 50 ejecuciones por entidad y número de semillas.

Tanto para el corpus de seminarios como para el de jobs se compararán los resultados obtenidos con los mostrados en el trabajo de Li et al. [2008]. En él se mejora un algoritmo de aprendizaje supervisado y se aplica a ambos corpus tanto con aprendizaje activo como sin él. Con aprendizaje activo concretamente reportan los mejores resultados obtenidos hasta la fecha sobre estos corpus (de los que no conocemos mejoras posteriores). Para ello utilizan como atributos los obtenidos de la herramienta ANNIE [Cunningham et al., 2013]: información sobre mayúsculas/minúsculas, tipo de token (*word, number* o *punctuation*), lema y *POS tagging* (aunque no queda claro en el artículo, consultados los autores indican que no usan clases semánticas provenientes de listados o de expresiones regulares en aprendizaje activo, aunque en el artículo sí indican que usan este tipo de atributos para aprendizaje supervisado). Comparan resultados usando como unidad de anotación el documento, fragmentos de 11 tokens de texto y secuencias de tokens de la entidad, utilizando siempre dos documentos completos anotados como semillas, que incluyen como coste de anotación (de nuevo, este punto no queda claro en el artículo y han debido ser consultados directamente los autores). Utilizan para entrenar el corpus completo y evalúan con aquello no utilizado durante el entrenamiento. Los resultados, a diferencia de los nuestros, son sobre todos los tipos de entidad de cada corpus a la vez, por lo que para compararlos en nuestro caso se sumarán los costes de anotación de cada tipo de entidad por separado. Esto supone una clara desventaja para nuestro método, puesto que la captura de cada tipo de entidad no cuenta con información sobre los restantes. Esta información en su caso es utilizada no sólo en el propio método de generación de patrones sino en una posterior fase de post-procesamiento, donde a cada entidad se le asigna el más probable de entre todos los tipos anotados en el corpus utilizando información acerca de la secuencia.

7.3.1. Seminarios

Efectividad vs coste de anotación

Los resultados en el corpus de seminarios muestran unas tasas de entre 0.39 y 0.452 de F-measure a costa de la validación de entre 194 y 884 ejemplos, incluyendo las semillas[26] (Tabla 28). De media, algo más de 5 tokens han de ser validados por cada ejemplo.

Tabla 28. Valores medios de efectividad (F-measure, precisión y recall) y coste (fragmentos de texto validados y su equivalente en tokens) resultantes en el corpus de seminarios para diferente número de semillas iniciales.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
F-measure	0.39±0.021	0.408±0.012	0.421±0.009	0.432±0.009	0.452±0.009	0.452±0.008
Precision	0.580±0.026	0.599±0.023	0.587±0.025	0.588±0.017	0.562±0.018	0.524±0.018
Recall	0.485±0.019	0.537±0.012	0.581±0.007	0.611±0.006	0.655±0.004	0.682±0.004
Ejemplos	194±11	249±10	323±10	423±10	643±11	884±13
Tokens	986±60	1287±59	1686±61	2244±69	3499±74	4906±87

En la Figura 23 y Figura 23 se muestra el resultado de las diferentes ejecuciones sobre el corpus. Como era de esperar, un mayor número de semillas mejora la efectividad, aunque también aumenta el número de ejemplos propuestos al usuario para validar (además de las propias semillas).

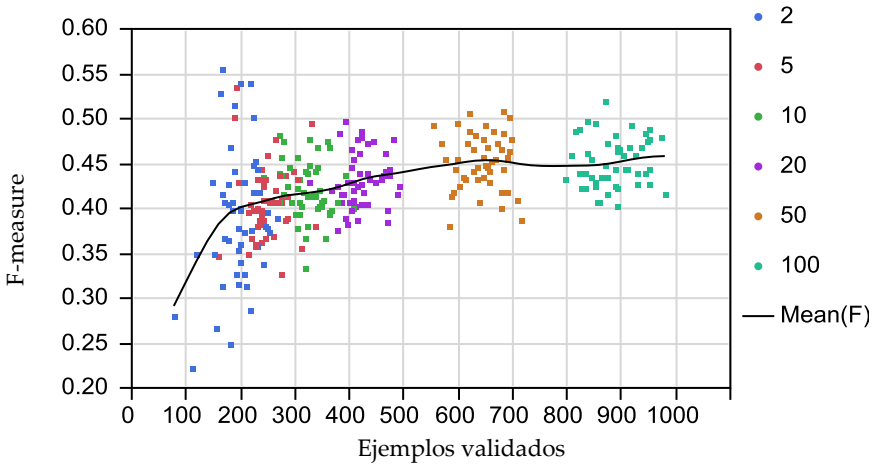


Figura 22. Relación de F-measure del conjunto de entidades y ejemplos validados obtenidos sobre el corpus de seminarios para cada una de las diferentes ejecuciones. La línea aproxima la F por número de ejemplos validados a partir de las medias obtenidas por número de semilla.

[26] En todos los casos los ejemplos y tokens mostrados incluyen las semillas. El número de tokens incluye además los contenidos en una ventana de 10 caracteres usada para la validación por parte del usuario.

Se observa además una importante variabilidad de la efectividad obtenida al utilizar tan sólo dos semillas iniciales. Esto muestra la dependencia de las semillas concretas que son seleccionadas. Sin embargo, ya a partir de 5 semillas, la variabilidad se reduce notablemente, y mantiene unos rangos de observaciones bastante estables, con variaciones de en torno a ±0.06 de F. El rango de observaciones del coste de anotación se mantiene bastante estable para distinto número de semillas, con un rango de entre 170 y 200 ejemplos.

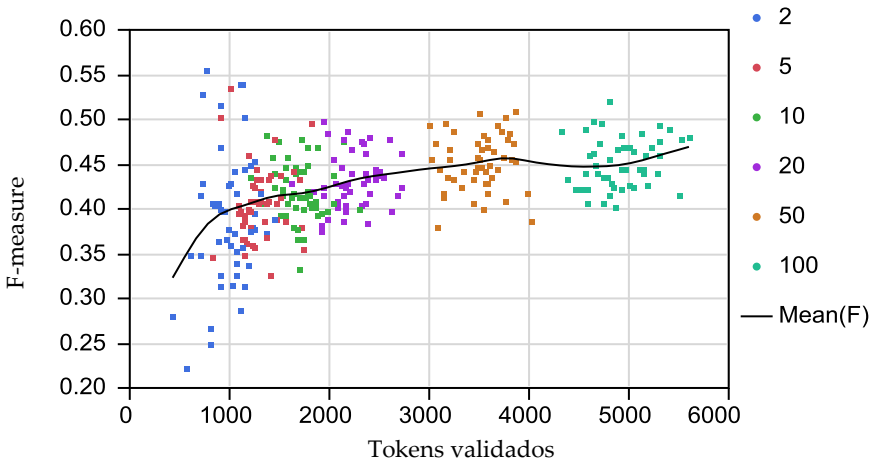


Figura 23. Relación de F-measure del conjunto de entidades y tokens validados obtenidos sobre el corpus de seminarios para cada una de las diferentes ejecuciones. La línea aproxima la F por número de tokens validados a partir de las medias obtenidas por número de semilla.

Precision y recall

En cuanto a las curvas de precision y recall (Figura 24) vemos que ante un aumento del número de semillas empeora la primera y mejora la segunda: las reglas tienden a ser más genéricas cuanto mayor es el número de ejemplos aportados. Como veremos posteriormente en este mismo epígrafe, esto ocurre fundamentalmente a causa de las entidades *stime* y *etime*. El punto de intersección se encuentra en poco más de 300 ejemplos anotados, incluyendo las semillas, lo que ocurre tomando alrededor de 10 semillas.

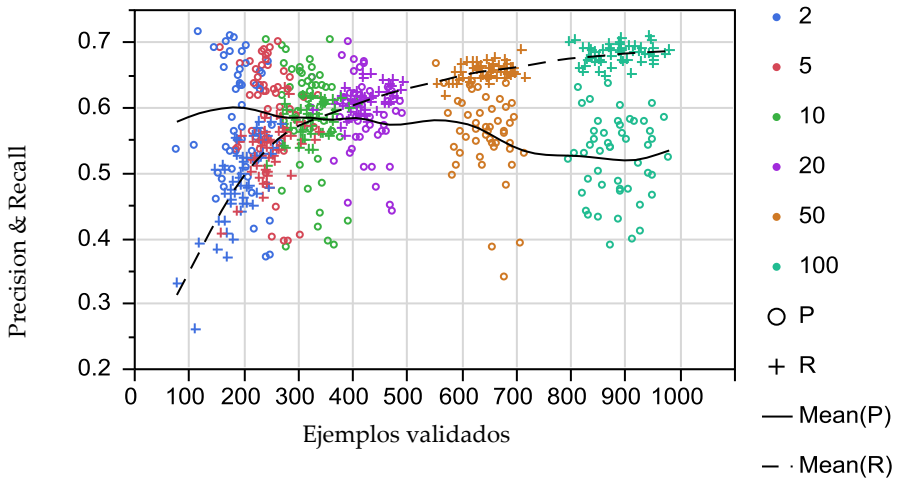


Figura 24. Relación entre la precision (P) y el recall (R) en función del número de ejemplos validados para cada número de semillas iniciales.

Fase I vs fase II

En la Figura 25 podemos apreciar cómo en este caso la fase II no parece aportar prácticamente mejoras sobre la efectividad de la fase I al utilizar menos de 50 semillas iniciales, y de hecho muestra peor comportamiento en ese rango. Sólo a partir de 50 semillas la Fase II parece alcanzar los resultados de la fase I e incluso mejorarlos en algunos casos. Sin embargo, estas ligeras mejoras en la fase II respecto a la fase I tienen un importante impacto negativo en el coste de anotación, que supone un incremento medio de 144 ejemplos a validar, un 48% más.

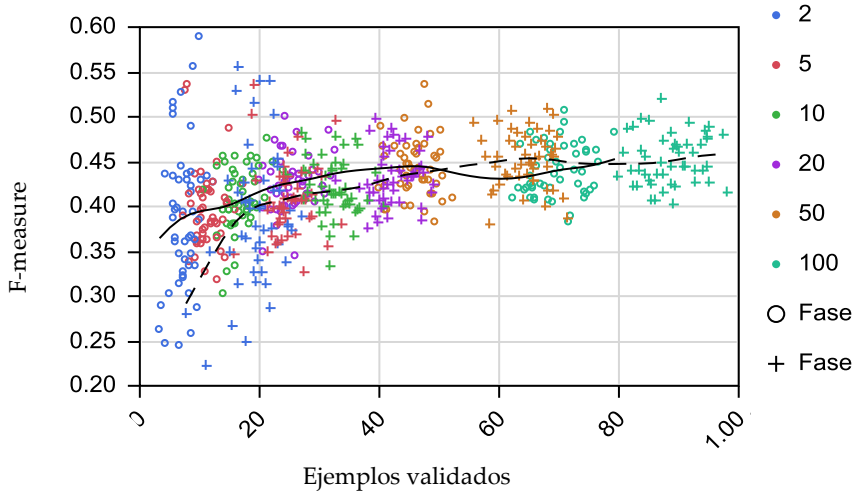


Figura 25. Relación entre la efectividad en términos de F-measure y coste de anotación en el corpus de seminarios para cada una de las fases. Los colores indican las semillas iniciales. La línea continua indica la aproximación de la F-measure en fase I, mientras que la discontinua corresponde a la fase II.

Comparativa

Para comparar los resultados con el trabajo de Li et al., en lugar de calcular la media del coste para cada ejecución sobre todos los tipos de entidad, sumamos estos costes. Esto nos da la suma de los costes para todas las entidades en esa ejecución y la F-measure media obtenida (Tabla 29). Hay que tener en cuenta en cualquier caso que este modo de comparación nos perjudica. Esto se debe a que en su caso cada entidad aporta información que puede contribuir a las restantes (de hecho, como veremos en la próxima sección, esto es lo que ocurre) y la anotación de unos tipos supone una anotación negativa para los restantes.

Los resultados pueden verse en la Figura 26 para la fase I y Figura 27 para la fase II. Se observa que en ambos casos la F-measure es menor entre un 0.05 y un 0.12 aproximadamente, aunque en la fase I los resultados son ligeramente mejores.

Tabla 29. Valores medios de F-measure y suma de los costes de anotación para todos los tipos de entidad.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
F-measure	0.39±0.021	0.408±0.012	0.421±0.009	0.432±0.009	0.452±0.009	0.452±0.008
Ejemplos total	777±11	996±10	1291±10	1691±10	2573±11	3535±13
Tokens total	3944±60	5148±59	6743±61	8976±69	13994±74	19623±87

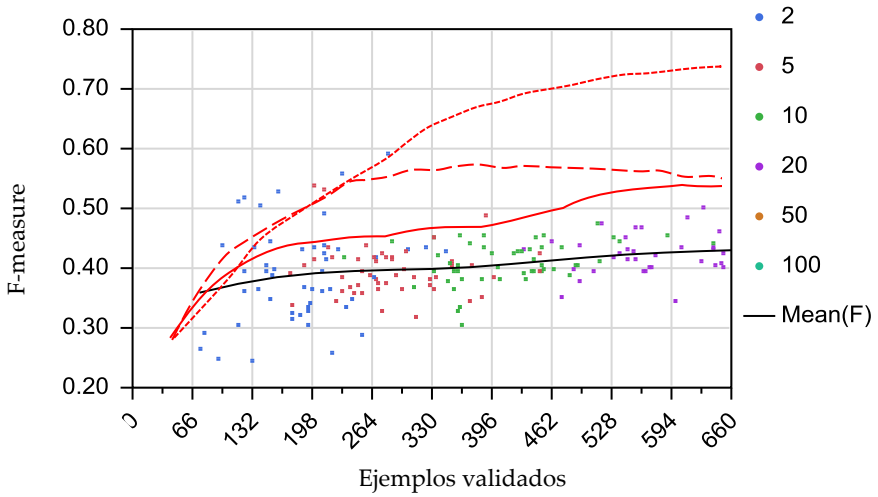


Figura 26. Comparación de la F-measure obtenida en la fase I sobre el corpus seminarios con el algoritmo presentado en [Li et al., 2008a] y utilizando como unidad de anotación los tokens (línea roja continua), fragmentos de 11 tokens (línea roja discontinua) y documentos (línea roja punteada).

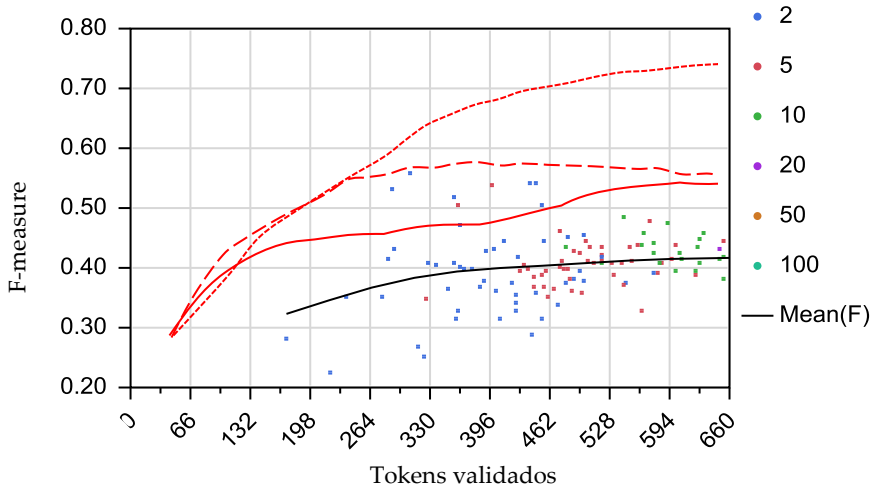


Figura 27. Comparación de la F-measure obtenida en la fase II sobre el corpus seminarios con el algoritmo presentado en [Li et al., 2008a] y utilizando como unidad de anotación los tokens (línea roja continua), fragmentos de 11 tokens (línea roja discontinua) y documentos (línea roja punteada).

Análisis por tipo de entidad

Si realizamos un examen más exhaustivo de los resultados por tipo de entidad (Figura 28) vemos que las entidades que peor se comportan son *etime* y *stime*, donde las tasas decrecen a medida que aumentamos el número de semillas. Por el contrario, la entidad *location* obtiene buenas tasas con relativamente pocos tokens, y en *speaker* se aprecia una tasa de crecimiento considerable al aumentar la cantidad de anotaciones.

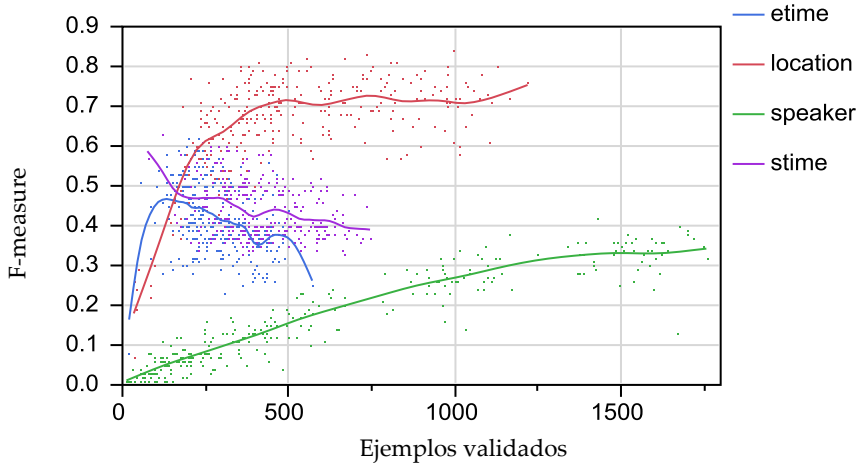


Figura 28. F-measure obtenida en la fase II en relación a los ejemplos validados por tipo de entidad.

Un análisis más detallado de *etime* y *stime* muestra que el problema radica en la precision (Figura 29). Estudiando los falsos positivos vemos que gran parte de las identificaciones

son efectivamente horas, pero no las etiquetadas como de inicio o fin de un seminario. De hecho, el 62.6% de las identificaciones en el caso de *stime* son horas, mientras que para *etime* esta tasa aumenta al 91.6% de identificaciones.

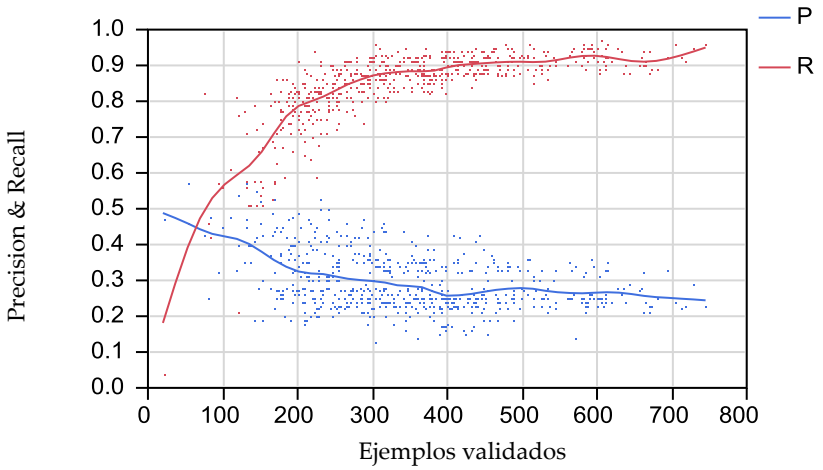


Figura 29. Precision y recall obtenidas en la fase II para los tipos *etime* y *stime*.

Estos resultados indican que la diferencia entre la hora de inicio y fin de un seminario no está basada en caracteres, y al parecer tampoco en los atributos utilizados. Al añadir más cantidad de semillas al algoritmo la confusión se incrementa, puesto que aumentan los tipos de patrones para capturar horas: las reglas creadas en la fase I son demasiado genéricas, y la fase II no consigue mejorar esta situación por no tener los atributos necesarios para discriminar unos tipos y otros de horas. Sí ayudaría sin embargo contar con atributos que procedieran de la detección de otras entidades: por ejemplo *etime* siempre ocurre tras *stime.* Muchos algoritmos de aprendizaje automático aprovechan esta información mediante el uso de cadenas de Markov. De este modo, son capaces de determinar la secuencia más probable entre diferentes tipos de entidades. En el caso de Li et al. aplican para ello una simulación del algoritmo de Viterbi en una fase de post-procesamiento. Esto en general resulta ventajoso, aunque afecta a la validez de los resultados cuando únicamente queremos capturar uno de los tipos de entidades, o cuando varía la secuencia de las entidades en los documentos de aplicación.

Para mostrar esta circunstancia hemos realizado dos experimentos: uno en el que para anotar uno de los tipos de entidad de hora se cuenta con información sobre el otro en forma de atributo, y otro en el que sustituimos ambos tipos por uno único.

Tabla 30. F-measure media y suma de ejemplos y tokens validados para todos los tipos de entidad por número de semillas, considerando atributos adicionales para *stime* y *etime.*

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
F-measure	0.473±0.025	0.483±0.015	0.512±0.012	0.534±0.012	0.561±0.015	0.562±0.013
Diferencia	+0.083	+0.075	+0.091	+0.102	+0.109	+0.11
Ejemplos total	454±31	647±30	897±32	1256±38	2083±39	3018±50
Diferencia	-323	-349	-394	-435	-490	-517
Tokens total	2416±191	3504±204	4896±214	6928±277	11655±288	17107±343
Diferencia	-1528	-1644	-1847	-2048	-2339	-2516

En el primero de los casos, al utilizar como atributo para *etime* las etiquetaciones de *stime* y a la inversa, se obtienen los resultados de la Tabla 30, donde vemos que de media por tipo de entidad la eficacia mejora entre 0.075 y 0.11 puntos. Además se reducen los costes de anotación entre 300 y más de 500 ejemplos según el número de semillas utilizadas.

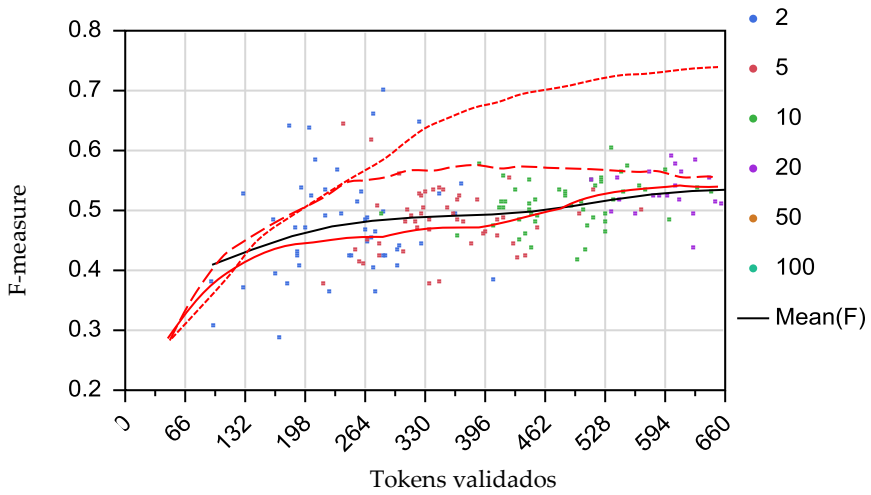


Figura 30. Relación entre los resultados de Li et al. para las diferentes unidades de anotación y los obtenidos al añadir como atributo la entidad *stime* para la identificación de *etime* y a la inversa.

Superponiendo estos resultados a los obtenidos en Li et al. (Figura 30) vemos que el comportamiento es muy similar, llegando a superarlo ligeramente al usar 5 semillas iniciales. Aunque estos resultados no son directamente comparables, sí dan idea de la influencia de contar con información sobre las restantes entidades durante el proceso de anotación. Esto puede resultar ventajoso en múltiples ocasiones, sin embargo estas técnicas no son aplicables a la captura independiente de cada tipo de entidad.

Para mostrar que el problema no radica en la captura de este tipo de entidad, sino en la falta de atributos para poder diferenciarlas se realiza un nuevo experimento, esta vez

sustituyendo ambos tipos por una única entidad, que llamaremos *time*. Para ello elaboramos manualmente una expresión regular (ver Código 27) sin falsos positivos y capaz de capturar una gran cantidad de indicaciones de hora existentes en el corpus, además de las entidades *etime* y *stime* (tan solo deja sin capturar 2 entidades de *etime* y 12 de *stime* que consisten únicamente en un número, frente a las 1415 entidades de ambos tipos anotadas) y ejecutamos el algoritmo sobre esta entidad, con idénticas condiciones a las anteriores.

```
\d\d?:\d\d?
(\s?(pm|PM|p\.m\.|P\.M\.|p\.m|P\.M|am|AM|a\.m\.|A.M\.|a\.m|A\.M|noon))
?
|
\d+\s?
(pm|PM|p\.m\.|P\.M\.|p\.m|P\.M|am|AM|a\.m\.|A\.M\.|a\.m|A\.M|noon)
```

Código 27. Expresión regular para la captura de indicaciones de hora en el corpus seminarios. La complejidad de la misma se debe a la necesidad de evitar la confusión con otras entidades numéricas.

Los resultados obtenidos (Tabla 31) muestran que el algoritmo es capaz de generar patrones para esta entidad con una F media de 0.924, con tan sólo dos semillas y una media de 344 ejemplos anotados, incluyendo las propias semillas.

Tabla 31. Relación de efectividad obtenida (F-measure, precision y recall) y coste (ejemplos validados y el equivalente en tokens) por número de semillas para la entidad *time.*

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
F-measure	0.92±0.008	0.918±0.008	0.929±0.007	0.923±0.01	0.932±0.009	0.921±0.012
Precision	0.958±0.01	0.946±0.012	0.949±0.013	0.929±0.018	0.934±0.017	0.903±0.022
Recall	0.886±0.008	0.893±0.008	0.910±0.005	0.919±0.005	0.932±0.004	0.943±0.004
Ejemplos	342±20	361±20	411±18	461±15	521±15	607±17
Tokens	1131±61	1190±73	1390±69	1548±56	1756±56	2056±62

Comparando con los resultados sobre el conjunto de entidades originales, y sumando los costes de anotación para todos los tipos de entidad (Tabla 32), en este caso tendríamos un aumento de la F-measure de entre 0.15 y 0.2, logrando tasas superiores a 0.50 de F-measure partiendo tan sólo de dos semillas. Se produce también una reducción en el coste total de más de 100 ejemplos con 2 semillas y más de 400 con 100, debido en buena medida a la reducción de los tipos de entidad.

Tabla 32. F-measure media y suma de ejemplos y tokens validados para los tipos *location*, *speaker* y *time* por número de semillas.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
F-measure	0.555±0.033	0.546±0.018	0.571±0.008	0.596±0.008	0.639±0.008	0.655±0.008
Diferencia	+0.165	+0.138	+0.15	+0.164	+0.187	+0.2
Ejemplos total	647±38	834±37	1103±37	1471±40	2231±39	3098±44
Diferencia	-130	-162	-188	-220	-342	-437
Tokens total	2693±202	3688±216	5107±224	7120±274	11433±282	16455±298
Diferencia	-1251	-1460	-1636	-1856	-2561	-3168

De nuevo, aunque estos resultados no son directamente comparables con los de otros trabajos sobre el corpus seminarios por haber cambiado las entidades objetivo, sí dan muestra de la utilidad de la metodología para la generación de gramáticas, con tasas de efectividad competitivas y a bajo coste de anotación en relación al necesario en otras aproximaciones. De hecho, incluso si consideramos el coste total de anotar todos los tipos de entidad en el corpus y lo superponemos a los resultados de Li et al., se observa que son superiores en hasta 0.12 puntos y se asemejan a los obtenidos utilizando como unidad de anotación fragmentos de 11 tokens, en lugar de tokens aislados (Figura 31).

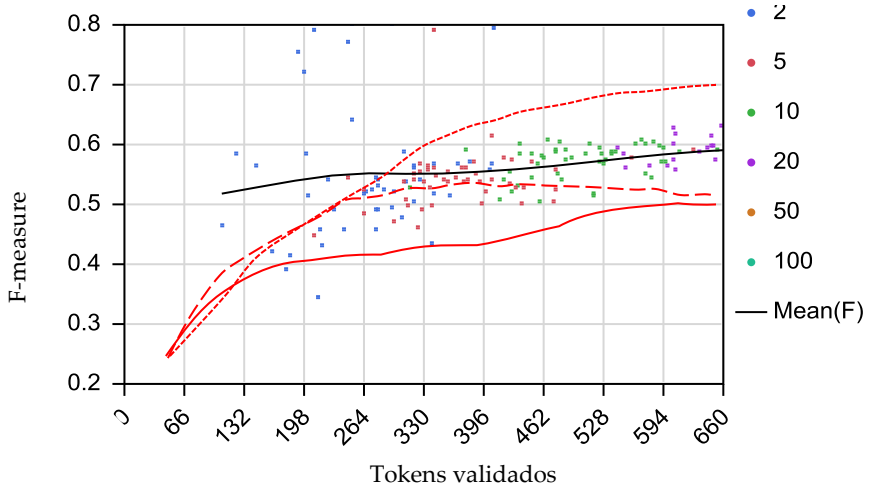


Figura 31. Relación entre la F-measure obtenida en la fase II sobre el corpus seminarios (sustituyendo los tipos *etime* y *stime* por *time*) con el algoritmo presentado en [Li et al., 2008a] y utilizando como unidad de anotación los tokens (línea roja continua), fragmentos de 11 tokens (línea roja discontinua) y documentos (línea roja punteada).

7.3.2. Jobs

Efectividad vs coste de anotación

Los resultados en el corpus jobs (Tabla 33) muestran unas tasas de entre 0.473 y 0.615 de F-measure media a costa de la validación de entre 88 y 521 ejemplos, incluyendo las semillas, con una media de casi 4 tokens por ejemplo validado (igual que en el caso anterior el recuento de número de tokens incluye también la ventana de 10 caracteres alrededor de los ejemplos propuestos al usuario). De nuevo se aprecia una clara diferencia en el coste de anotación al usar un número u otro de semillas, aumentando a medida que aumentamos también el número de semillas iniciales.

Tabla 33. Valores medios de efectividad (F-measure, precision y recall) y coste (fragmentos de texto validados y tokens equivalentes) por tipo de entidad en el corpus jobs para diferente número de semillas iniciales.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
F-measure	0.473±0.014	0.557±0.01	0.594±0.008	0.608±0.008	0.609±0.006	0.615±0.006
Precision	0.754±0.017	0.719±0.013	0.696±0.012	0.659±0.014	0.626±0.011	0.615±0.009
Recall	0.385±0.011	0.524±0.011	0.593±0.008	0.646±0.006	0.693±0.006	0.716±0.006
Ejemplos	88±3	159±5	222±4	293±4	413±5	521±5
Tokens	305±13	611±23	862±22	1160±21	1665±25	2115±25

En relación al corpus de seminarios se observan no sólo mejores tasas de efectividad sino a menor coste de anotación, con un crecimiento más pronunciado de la F-measure entre 2 y 10 semillas (Figura 32 y Figura 33). El valor ideal parece encontrarse en 10 semillas, lográndose tasas de casi el 0.60 de F-measure con un coste de menos de poco más de 200 anotaciones, incluyendo las semillas.

Los rangos de observaciones son similares a los obtenidos en el corpus seminarios. Como en aquél caso, al aumentar el número de semillas iniciales se obtiene menor variabilidad en la efectividad, pero ligeramente más variabilidad en el coste de anotación. La variabilidad de la efectividad se mantiene alrededor de ±0.06 a partir de 10 semillas. La variabilidad del coste de anotación no llega a los 100 ejemplos y es por tanto menor que en el corpus seminarios.

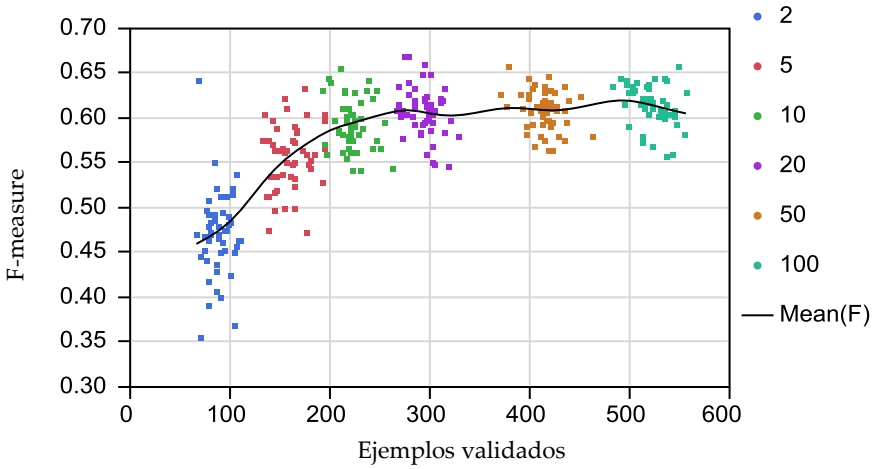


Figura 32. Relación de F-measure y ejemplos validados sobre el corpus jobs. Los colores indican el número de semillas iniciales, y la línea es una aproximación sobre la media de las ejecuciones por número de semillas.

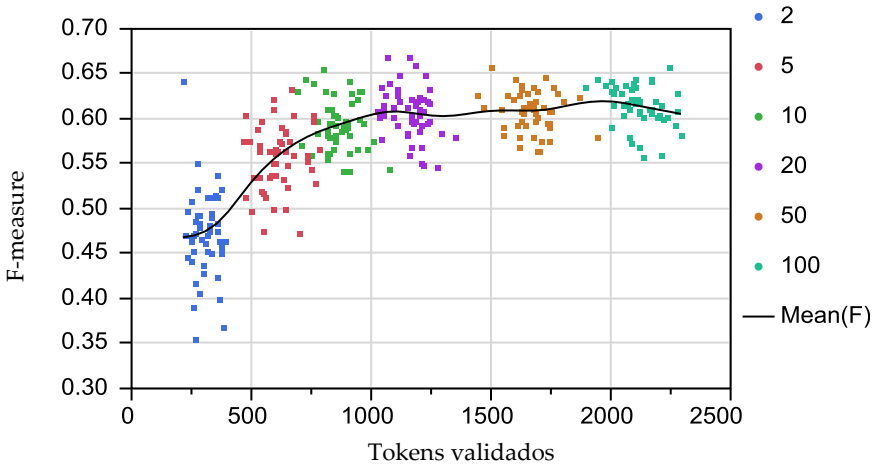


Figura 33. Relación de F-measure y tokens validados sobre el corpus jobs. Los colores indican el número de semillas iniciales, y la línea es una aproximación sobre la media de las ejecuciones por número de semillas.

Precision y recall

Las gráficas de precision y recall (Figura 34) muestran también comportamientos similares a los observados para el corpus seminarios, aunque en este caso la caída de la precision es menor (en aquel caso ya vimos que se explicaba por los tipos *etime* y *stime).* Ambas curvas parecen converger a partir de 400 anotaciones y, como en el caso de seminarios, el punto de intersección vuelve a situarse alrededor de los 300 ejemplos anotados, que en este caso corresponde a la elección de 20 semillas iniciales.

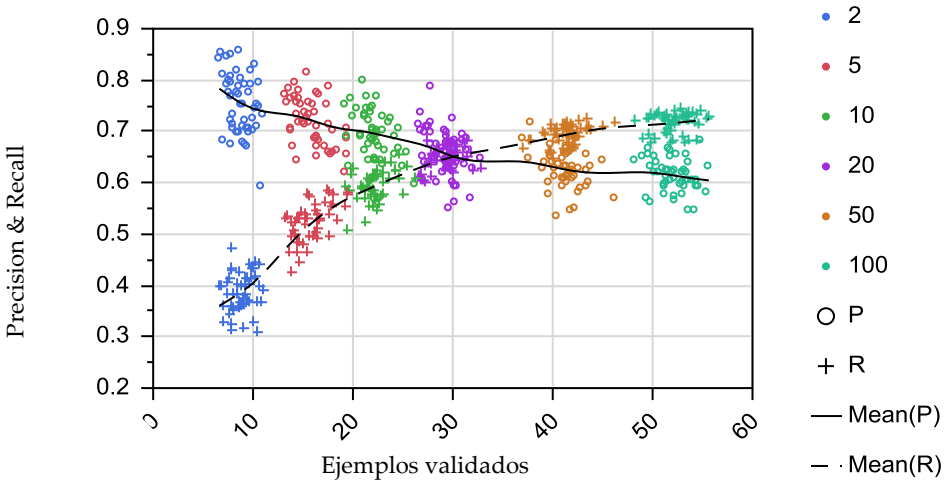


Figura 34. Relación entre la precision (P) y el recall (R) en función del número de ejemplos validados para cada número de semillas iniciales, mostradas en código de colores.

Fase I vs fase II

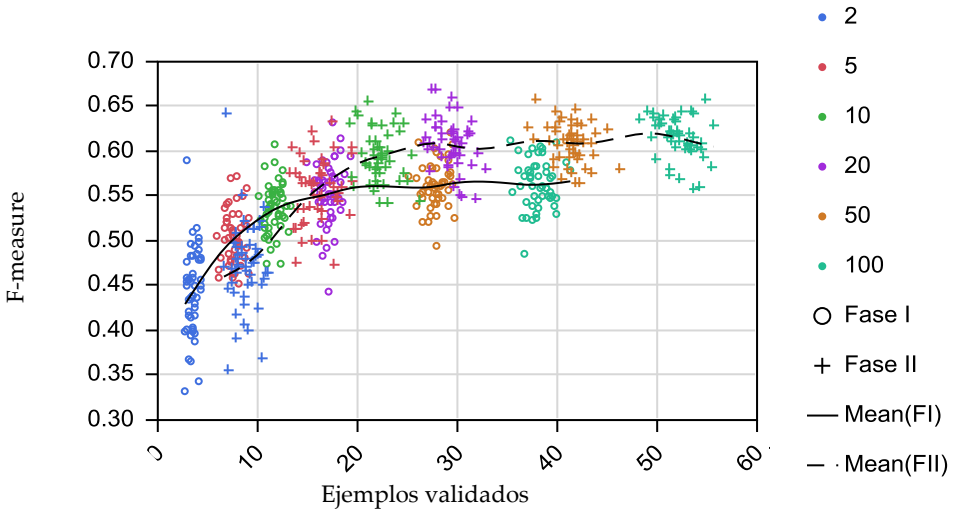


Figura 35. Relación entre la efectividad en términos de F-measure y coste de anotación en el corpus jobs para cada una de las fases. Los colores indican las semillas iniciales. En línea continua se representa la aproximación de la F-measure de la fase I, en discontinua la de la fase II.

En cuanto al comportamiento de cada una de las fases (Figura 35), para este corpus sí se aprecia la influencia positiva de la fase II, que logra superar a la fase I a partir de aproximadamente 150 ejemplos anotados, lo que ocurre ya con al menos 5 semillas. La posición respecto al eje X de los conjuntos de un mismo color (mismo número de semillas iniciales) también nos indica que, como es lógico, la aplicación de la fase II conlleva un coste de anotación, que es mayor cuanto mayor es el número de semillas iniciales: una

mayor cantidad de ejemplos favorece la creación de reglas potenciales que deberán ser validadas por el usuario.

Comparativa

Para comparar los resultados obtenidos con el trabajo de Li et al. como hicimos para el caso de seminarios, en primer lugar realizamos los cálculos eliminando la entidad *phone*, de modo que el conjunto de entidades evaluadas sea exactamente el mismo. Los resultados obtenidos de efectividad y coste pueden verse en la tabla Tabla 35. este caso además de sumar los costes de cada tipo de entidad,

Tabla 34. Valores medios de efectividad (F-measure, precision y recall) y coste (fragmentos de texto validados y tokens equivalentes) por tipo de entidad en el corpus jobs (eliminando *phone*) para diferente número de semillas iniciales.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
F-measure	0.442±0.014	0.534±0.01	0.573±0.008	0.587±0.008	0.589±0.006	0.6±0.006
Precision	0.741±0.018	0.706±0.014	0.682±0.013	0.643±0.014	0.608±0.011	0.596±0.01
Recall	0.353±0.011	0.499±0.011	0.571±0.009	0.627±0.007	0.676±0.006	0.701±0.006
Ejemplos	88±3	159±5	222±4	293±4	413±5	521±5
Tokens	305±13	611±23	862±22	1160±21	1665±25	2115±25

Como en el caso seminarios, para realizar la comparación sumamos entonces los costes de reconocer cada tipo de entidad (Tabla 35).

Tabla 35. Valores medios de F-measure y suma de los costes para el reconocimiento de todos los tipos de entidad.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
F-measure	0.442±0.014	0.534±0.01	0.573±0.008	0.587±0.008	0.589±0.006	0.6±0.006
Ejemplos total	1406±3	2673±5	3758±4	5022±4	7148±5	9050±5
Tokens total	4821±13	10252±23	14624±22	19917±21	28844±25	36815±25

La gráfica comparativa (Figura 36) muestra que ya tan solo con dos semillas obtenemos mejores tasas, tanto utilizando tokens como fragmentos de tokens. Nuestro método parte de una F de 0.44 tan sólo con dos semillas y alcanza una F de 0.60 con 100 semillas, mientras en su caso, utilizando como unidad de anotación los tokens, sólo es posible alcanzar una F máxima de 0.37, que aumenta a 0.41 al utilizar fragmentos de texto en lugar de tokens. Esto supone un aumento de efectividad de entre el 36% y el 62% (considerando tokens) y de entre el 36% y el 46% (considerando fragmentos de texto). Incluso utilizando

como unidades de anotación documentos completos vemos que nuestro método supera su efectividad con cualquier número de semillas con que se realizaron los experimentos. Por ejemplo, mientras ellos necesitan unos 31500 tokens (unos 100 documentos[27] aproximadamente) para llegar a 0.57 de F, nuestro método alcanza esas tasas con 10 semillas y un total de 14624 tokens anotados (el equivalente a unos 47 documentos), lo que supone un ahorro del 54% en anotación. Incluso si consideráramos que para identificar cada una de las semillas aportadas es necesario anotar un documento completo, obtendríamos un coste total aproximado de alrededor de 57 documentos, lo que supone un ahorro del 43%.

Por otro lado, aunque nuestro algoritmo no llega a los valores en torno a 0.65 de F-measure que ellos alcanzan (de nuevo, utilizando como unidad de anotación el documento, lo que además de aportar múltiples ejemplos negativos, supone que en cada anotación se anotan ejemplos de prácticamente todas las entidades a la vez), esto les requiere en torno a 72000 tokens, lo que supone la anotación de unos 230 documentos de los 300 que forman el corpus (el 77% del total).

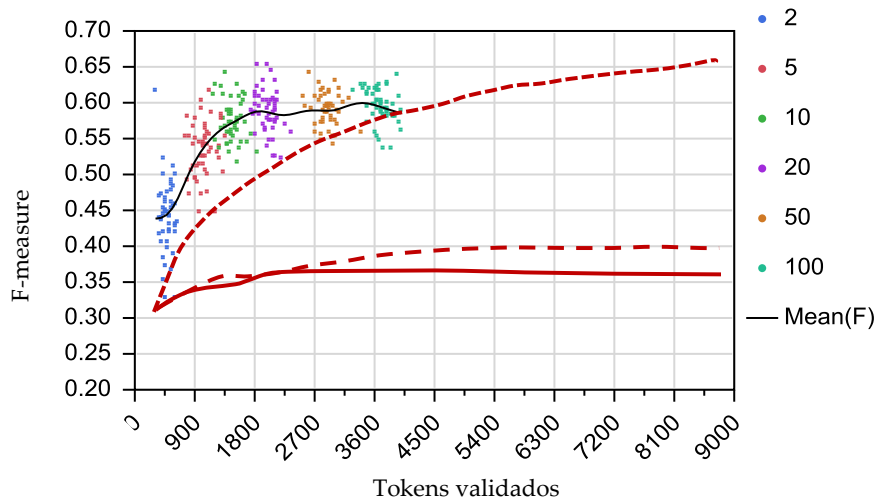


Figura 36. Comparación de la F-measure obtenida en la fase II sobre el corpus jobs con el algoritmo presentado en [Li et al., 2008a] y utilizando como unidad de anotación los tokens (línea roja continua), fragmentos de 11 tokens (línea roja discontinua) y documentos (línea roja punteada).

[27] 312.786 tokens de media por documento del corpus Jobs

Análisis por tipo de entidad

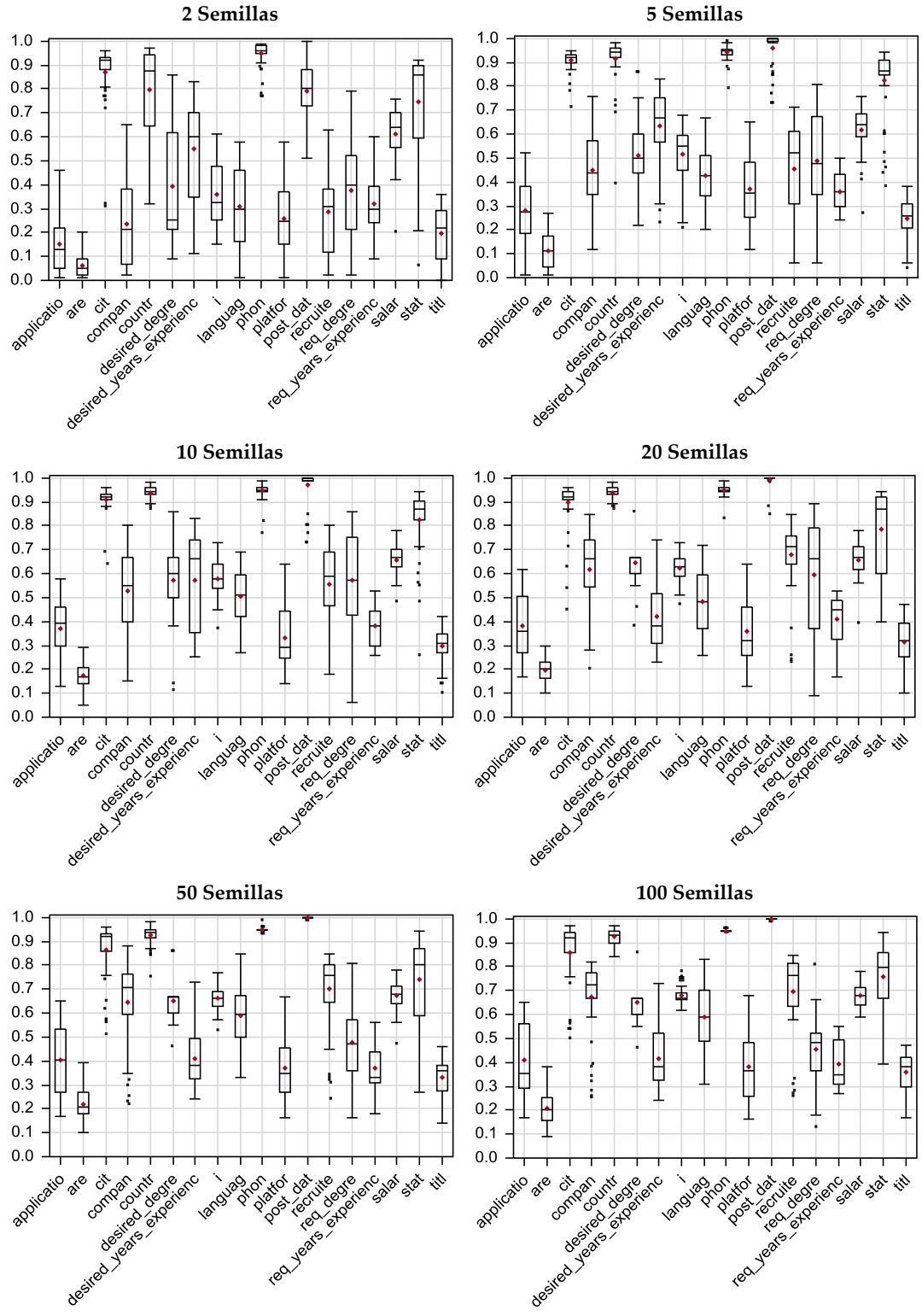


Figura 37. Diagramas de caja y medias (rombos) obtenidas de la F-measure por entidad en el corpus jobs.

En cuanto a los resultados por tipo de entidad nos encontramos con que son variables, especialmente al utilizar tan solo dos semillas iniciales (Figura 37). Aun así, tan solo con esas dos semillas es posible generar gramáticas con tasas de F-measure superiores a 0.6 de media para 6 de los 18 tipos de entidad: *city*, *country*, *phone*, *post_date*, *salary* y *state*. Con 20 semillas esas tasas aumentan y además otras 4 entidades alcanzan tasas de 0.6 o superiores: *company*, *desired_degree*, *id*, y *recruiter*. Nuestra metodología, por tanto, ha sido capaz de reconocer 10 de las 18 entidades en el corpus con tasas iguales o superiores a 0.6 sin corpus anotados previos y utilizando tan sólo atributos simples. La incorporación de atributos más potentes que los utilizados aquí, como por ejemplo categorías semánticas u otras entidades previamente reconocidas, no supone ningún cambio en la metodología descrita, y ayudará a elevar estas tasas, aunque esto se plantea como trabajo futuro.

7.4. Tarea A. Generación de gramáticas IEG: efectividad y coste con mínimo número de semillas

Dado que las nuevas entidades validadas durante el proceso de aprendizaje podrían utilizarse para alimentar un nuevo proceso de aprendizaje capaz de mejorar los patrones, se ha diseñado un experimento para comprobar la relación eficacia/coste de anotación en este caso.

Para ello realizamos el experimento utilizando tan solo dos semillas iniciales, de modo que el coste de localización de semillas sea mínimo. Se comienza un proceso de aprendizaje de patrones utilizando estas dos semillas iniciales. Una vez concluído, comenzamos un nuevo proceso de aprendizaje tomando como semillas las entidades validadas durante el primer proceso. Se medirán entonces las tasas de efectividad obtenidas tras finalizar este segundo proceso y se considerarán como costes de anotación los ejemplos validados tanto durante el primer proceso como durante el segundo, además de las propias semillas iniciales.

La configuración del experimento será idéntica al anterior, con diez ejecuciones distintas variando la selección aleatoria de las dos semillas iniciales, y el uso de las mismas cinco divisiones diferentes del corpus en entrenamiento y test ya utilizadas en el caso anterior. En total, el número de ejecuciones para cada corpus ha sido:

Jobs: 18 $tipos\ de\ NE\ \times 5\ nº\ corpus\ \times 10\ selección\ semillas = 900$

Seminarios: 4 $tipos\ de\ NE\ \times 5\ nº\ corpus\ \times 10\ selección\ semillas = 200$

resultando así 50 ejecuciones por tipo de entidad.

7.4.1. Seminarios

Los resultados obtenidos muestran una F-measure media de 0.406, con un coste de anotación medio de 711 ejemplos (incluyendo también las dos semillas iniciales), y 3660 tokens. En la Tabla 36 y Figura 38 podemos observar estos datos comparados con los obtenidos en la Sección 7.3 al considerar cualquier número de semillas iniciales. Se aprecia un comportamiento en cuanto a efectividad similar al logrado al utilizar cinco semillas iniciales, aunque con un coste ligeramente superior al requerido cuando usamos 50 semillas iniciales.

Tabla 36. Resultados al aplicar un único proceso con un número variable de semillas iniciales o dos procesos consecutivos partiendo de 2 semillas iniciales.

	Semillas						**2 semillas**
	2	**5**	**10**	**20**	**50**	**100**	**/ 2 procesos**
F-measure	0.39	**0.408**	0.421	0.432	0.452	0.452	**0.406**±0.014
Precision	0.580	0.599	0.587	0.588	0.562	0.524	0.558±0.023
Recall	0.485	0.537	0.581	0.611	0.655	0.682	0.6±0.01
Ejemplos	194	249	323	423	**643**	**884**	**711**±34
Tokens	986	1287	1686	2244	**3499**	**4906**	**3660**±209

Los resultados por tanto suponen una mejora respecto de la efectividad al utilizar un único proceso con dos semillas iniciales, aunque aumentan el coste de anotación en más del triple. Analizando las medidas de precision y recall alcanzados con el mismo coste de anotación en un solo proceso vemos que mientras que ese coste de anotación corresponde al uso de entre 20 y 50 semillasm y que mientras que la precision es similar, el recall es inferior, lo que hace que la efectividad descienda.

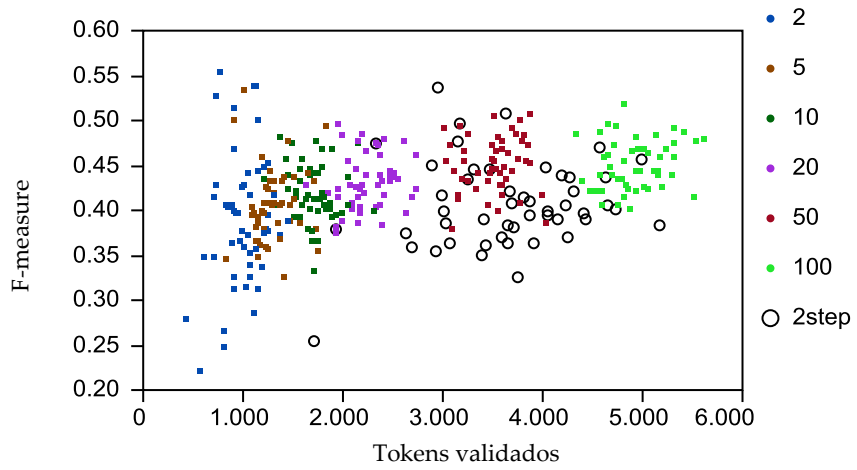


Figura 38. Relación de efectividad y coste al aplicar un único proceso con un número variable de semillas iniciales (Sección 7.3.1) o dos procesos consecutivos partiendo de 2 semillas iniciales.

Este problema puede reducirse de nuevo sustituyendo las entidades *stime* y *etime* por una única entidad *time* como explicamos en la Sección 7.3.1. Los resultados pueden apreciarse en la Tabla 37: en este caso la efectividad al utilizar tan sólo dos semillas con dos procesos es igual a la obtenida al uilizar 20 semillas iniciales con uno, aunque el coste supera al correspondiente a esas 20 semillas en un único proceso.

Tabla 37. Resultados al aplicar un único proceso con un número variable de semillas iniciales o dos procesos consecutivos partiendo de 2 semillas iniciales con las entidades *time*, *speaker* y *location*.

	Semillas						**2 semillas**
	2	**5**	**10**	**20**	**50**	**100**	**/ 2 procesos**
F-measure	0.555	0.546	0.571	**0.596**	0.639	0.655	**0.596**±0.022
Precision	0.908	0.898	0.893	0.899	0.881	0.838	0.894±0.027
Recall	0.433	0.472	0.511	0.538	0.585	0.617	0.523±0.011
Ejemplos	216	278	368	**490**	**744**	1033	**604**±35
Tokens	898	1229	1711	**2373**	**3811**	5485	**2805**±222

7.4.2. Jobs

Los resultados para el corpus jobs muestran una F-measure media de 0.733 con un coste de la anotación de 323 ejemplos de media en conjunto para ambos procesos. En la Tabla 38 y la Figura 39 se contrastan los resultados con los obtenidos utilizando un único proceso. La efectividad con dos procesos consecutivos y dos semillas iniciales son superiores a los obtenidos utilizando tan sólo un proceso con el mismo número de semillas y son similares a los obtenidos con 5 semillas. El coste, igual que con el corpus seminarios, prácticamente

se triplica, equiparándose al necesario al utilizar un proceso con entre 20 y 50 semillas iniciales. Igual que en aquel caso (y de forma más fiable al tratarse de 18 tipos de entidad frente a las 4 del corpus seminarios) es el recall lo que hace disminuir la eficacia, ya que para igual coste de anotación en un único proceso tendríamos una precision algo menor pero un recall más elevado (Figura 34).

Tabla 38. Jobs: resultados al aplicar un único proceso con un número variable de semillas iniciales o dos procesos consecutivos partiendo de 2 semillas iniciales.

	Semillas						2 semillas
	2	**5**	**10**	**20**	**50**	**100**	**/ 2 procesos**
F-measure	0.473	**0.557**	0.594	0.608	0.609	0.615	**0.552**±0.011
Precision	0.754	0.719	0.696	0.659	0.626	0.615	**0.733**±0.014
Recall	0.385	0.5241	0.593	0.646	0.693	0.716	**0.504**±0.012
Ejemplos	88	159	222	**293**	**413**	521	**323**±11
Tokens	305	611	862	**1160**	**1665**	2115	**1195**±42

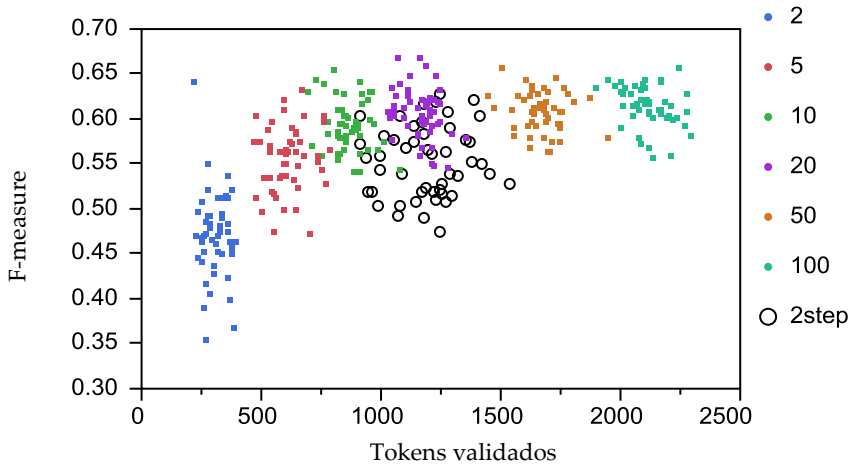


Figura 39. Jobs: relación de efectividad y coste al aplicar un único proceso con un número variable de semillas iniciales (Sección 7.3.2) o dos procesos consecutivos partiendo de 2 semillas iniciales.

7.5. Tarea A. Generación de expresiones regulares

La fase I de la metodología presentada genera expresiones regulares habituales sobre las entidades, con los caracteres representados como Unicode o agrupaciones de Unicode.

Estos resultados ofrecen la ventaja de ser aún más fáciles de gestionar que las gramáticas IEG, puesto que no requieren ningún tipo de adaptación para su edición y aplicación en otras herramientas (salvo la requerida para la adaptación al lenguaje de expresiones regulares de la plataforma que vaya a utilizarse). La mayor parte de las entidades analizadas en ambos corpus no presentan patrones de este tipo (e.g. *area, plataforma, speaker* etc.). Pero algunas excepciones son los tipos de entidad *time* en el corpus de seminarios, y *phone* y *post_date* en el corpus jobs.

Se ha diseñado un experimento que permite apreciar la potencia de la metodología ante este tipo de entidades, al tiempo que se pondrá en práctica el uso de semillas externas al corpus. Para ello, se ejecutará únicamente la fase I con las cinco divisiones del corpus utilizadas en los experimentos anteriores, y seleccionando aleatoriamente tres semillas de las recogidas en la Tabla 39. Ninguna de estas semillas, a las que llamaremos "semillas externas", pertenece al corpus sobre el que se aplica el algoritmo. Además, las entidades que aparecen en rojo presentan una estructura que no coincide con ninguna de las encontradas en el corpus (por ejemplo los meses de las entidades de tipo *post_date* siempre aparecen formados por tres letras, con la primera en mayúscula). Los resultados obtenidos se compararán con los logrados cuando las semillas forman parte del propio corpus de entrenamiento, a las que llamaremos "semillas propias". De este modo se pone a prueba la capacidad de la metodología presentada para generar expresiones regulares utilizando semillas externas que además no necesariamente siguen el patrón de las que se pretende reconocer, lo que es de esperar que ocurra si no se realiza un análisis previo del corpus de entrada. En cada caso (semillas externas o propias), la selección de las semillas se realiza de forma aleatoria. El número de ejecuciones será entonces:

Jobs: $2\ tipos\ de\ NE\ \times 5\ nº\ corpus\ \times 2\ tipo\ semillas\ \times 10\ selección\ semillas = 200$

Seminarios: $1\ tipo\ de\ NE\ \times 5\ nº\ corpus\ \times 2\ tipo\ semillas\ \times 10\ selección\ semillas = 100$

resultando 50 ejecuciones por tipo de entidad y tipo de semillas utilizadas (propias o externas).

Tabla 39. Semillas utilizadas para los tipos de entidad *time, phone* y *post_date.* No se encuentran en el corpus, y las que están en negrita además presentan patrones diferentes a las que se encuentran en él.

Time	Phone	Post_date
21.55 AM	**632 458 666**	23 Apr 2010
5am	**958-523-ASCI**	**7 March 1977**
3.45 pm	91-624-9115	25 Feb 75
08:45 A.m.	**916249-1155**	**14 january 2033**
05:55 pm	**632/945 7788**	14 May 85
5 a.m	(821)444-8241	**1 jun 85**
03:44 p.m.	**5554789632**	**28 July 2000**
14h.	**922 5554189**	**15 August 1989**
11:55	**698/458/2148**	**18 september 2020**
13h.	(452)954-3377	**9 oct 2001**

La Tabla 40 muestra los resultados medios obtenidos utilizando semillas propias del corpus de entrenamiento. Se alcanzan cotas de efectividad de entre 0.8 y 1, tanto en precision y recall como en F-measure, con un coste para el usuario de tan solo 20 ejemplos a validar para *post_date* y en torno a 53-54 para *time* y *phone,* incluyendo en todos los casos las propias semillas.

Tabla 40. Medias e intervalos de confianza con tres semillas propias.

	Time	Phone	Post_date
F-measure	0.796±0.027	0.93±0.016	0.858±0.037
Precision	0.824±0.028	0.922±0.018	1
Recall	0.77±0.026	0.939±0.017	0.771±0.054
Ejemplos	54±4	53±3	20±2
Tokens	284±23	300±16	95±9

Si en lugar de semillas contenidas en los corpus utilizamos aleatoriamente tres semillas de la Tabla 39, obtenemos los resultados que aparecen en la Tabla 41. En este caso las expresiones regulares generadas tienen una efectividad media de entre 0.7 y 0.8 de F, con valores de precision y recall de entre 0.7 y 0.9. El coste de anotación para la generación de estas expresiones ronda los 50-60 ejemplos anotados, incluyendo las tres semillas iniciales externas.

La diferencia entre los resultados con semillas propias y externas muestra una disminución de precision, recall y F-measure al utilizar semillas externas. La F disminuye entre 0.10 y 0.13 aproximadamente, provocada en mayor medida por la disminución de la precision en el caso de los tipos de entidad *phone* y *time.* En cuanto al coste, el aumento es muy ligero para *phone*, de tan sólo 3 ejemplos de media, algo mayor, con una media de 11

ejemplos anotados de diferencia para *time* y mayor, con 32 ejemplos anotados más para *post_date*.

Tabla 41. Medias e intervalos de confianza con tres semillas externas.

	Time	**Phone**	**Post_date**
F-measure	0.691±0.048	0.8±0.063	0.737±0.08
Precision	0.692±0.057	0.77±0.056	0.916±0.052
Recall	0.717±0.046	0.842±0.071	0.682±0.087
Ejemplos	65±6	56±11	52±5
Tokens	355±32	291±57	308±38

En definitiva, las tasas muestran muy buenos resultados para el caso de semillas propias, logrando a bajo coste de anotación generar expresiones regulares con tasas muy elevadas de efectividad. Pero también el uso de semillas externas da buenos resultados, con tasas superiores al 0.7 de F y alrededor de 55 anotaciones, con las ventajas que ello supone: independencia entre semillas y corpus de entrada, y reducción del coste de anotación para el usuario al evitar la revisión del corpus para la elección de las semillas. La metodología, gracias al proceso de aprendizaje activo, muestra además robustez ante semillas con estructuras diferentes a las contenidas en el corpus y que podrían inducir a errores.

7.6. Tarea B. Anotación de corpus

Los resultados presentados en las secciones 7.3, 7.4 y 7.5 muestran la capacidad de los patrones generados para adaptarse a partes no vistas en un mismo corpus. No resultaría válido en ese caso obtener como patrones por ejemplo cada una de las entidades reconocidas en el corpus de entrenamiento a modo de listado, lo que sería un ejemplo extremo de *overfitting*. Sin embargo este ejemplo extremo en ocasiones puede resultar de interés. Los patrones, aunque poco extrapolables a otros corpus, pueden ayudar a reducir los costes de anotación del propio corpus. Este tipo de técnicas son de especial interés en el ámbito de la evaluación, donde podrían ayudar en la creación de recursos anotados que, como vimos en el Capítulo 4, permitieran variar los tipos de entidad en las diferentes convocatorias de los foros.

En esta tarea tiene especial importancia el recall conseguido. Una herramienta no nos servirá para anotar todo un corpus si sólo nos ayuda a reconocer una pequeña porción de las entidades que queremos anotar, aunque sus tasas de precision sean elevadas. Para medir la relación entre la proporción de entidades del corpus reconocidas (recall) y el coste de anotación, se ha incluido en este último no sólo el coste de generación de los patrones sino además el coste de analizar cada uno de los fragmentos de texto recuperados por los patrones obtenidos. De este modo se mide el coste que supondría anotar sin errores el corpus (precision=1).

Se realizarán los mismos experimentos que para la generación de patrones IEG, contando con los mismos atributos y número de semillas, pero sin dividir el corpus en entrenamiento y test. El número de ejecuciones será entonces:

Jobs: 18 $tipos\ de\ NE\ \times 6\ nº\ semillas\ \times 10\ selección\ semillas = 1080$

Seminarios: 4 $tipos\ de\ NE\ \times 6\ nº\ semillas\ \times 10\ selección\ semillas = 240$

resultando 10 ejecuciones por entidad y número de semillas.

7.6.1. Seminarios

Los resultados obtenidos de media para el corpus de seminarios pueden verse en la Tabla 42. En ella se muestra el coste de generación de los patrones sumado al coste de validación de los fragmentos de texto que reconocen los patrones. El total de estos costes se muestra en número aproximado de documentos que supone anotar (de media tenemos 239.012 tokens/documento) y porcentaje del corpus que representan.

Tabla 42. Relación Recall y coste de anotación medios para el corpus Seminarios.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
Recall	**0.489**	**0.565**	**0.593**	**0.661**	**0.725**	**0.788**
Coste Generación (identificaciones)	212	269	359	472	713	993
Coste Generación (tokens)	1054	1363	1854	2449	3846	5552
Coste Validación (identificaciones)	1050	1038	1162	1298	1342	1527
Coste Validación (tokens)	5487	5425	5814	6781	7014	7980
Coste total (tokens)	6541	6788	7668	9230	10860	13532
Coste total aprox. (documentos)	27	28	32	39	45	57
Coste total aprox. (% corpus)	**5.6**	**5.8**	**6.6**	**8**	**9.3**	**11.8**

En el corpus seminarios vemos que a partir de 5 semillas podemos anotar de media más de la mitad de las entidades (de media por tipo de entidad), anotando el equivalente aproximado a 28 documentos (un 5.8% del corpus). Es posible llegar anotar más de un 70% de las entidades utilizando al menos 50 semillas iniciales y anotando el equivalente aproximado a entre 45 y 57 documentos, que aún supone menos del 12% del corpus.

Si en lugar de contabilizar como esfuerzo de anotación únicamente los tokens de las semillas, ya incluidos en los costes de generación, consideramos que cada semilla supone la anotación de un documento completo (es frecuente encontrar una entidad de cada tipo en cada documento), entonces tendríamos los resultados de la Tabla 43. En ella vemos que aún en el peor caso (con 100 semillas), el texto anotado en total no supera el 32% del corpus para el reconocimiento de casi el 79% de cada tipo de entidad.

Tabla 43. Relación Recall y coste de anotación medios, considerando que el coste de obtener una semilla equivale a la anotación de un documento completo.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
Recall	0.489	0.565	0.593	0.661	0.725	0.788
Coste total aprox. (% corpus)	6	6.8	8.7	12	19.4	31.8

Un análisis más detallado por tipo de entidad de estos resultados (Figura 40) muestra que son muy dispares. En el caso del tipo *speaker* vemos que, aunque la precision es muy alta, nos costaría alrededor de 1600 ejemplos anotados, empezando con 100 semillas, lograr tasas de 0.5 de recall. Esto supondría anotar el equivalente a unos 35 documentos para obtener la mitad de las entidades de ese tipo en el corpus. Estos ratios mejoran para el tipo *location,* para el que, comenzando con 50 semillas y con un coste de tan sólo aproximadamente 17 documentos anotados podríamos obtener tasas de 0.8 tanto en precision como en recall.

Pero la utilidad de esta metodología para la anotación de corpus se aprecia en mayor medida para los tipos *stime* y *etime.* En la Tabla 44 vemos que con tan sólo 5 semillas iniciales y menos de 400 ejemplos validados en la generación de patrones, podemos lograr tasas de recall de alrededor de 0.8 aunque a costa de una precision de algo más de 0.3 (provocada por la confusión ya comentada entre horas de uno y otro tipo). Incluyendo el coste de validación de las identificaciones realizadas por esos patrones, en total nos

supondría anotar el equivalente a 30 y 63 documentos para *etime* y *stime* respectivamente. Es decir, con un 6% del corpus podemos anotar el 77% de las entidades de tipo *etime*, mientras que en el caso de *stime* la tasa sube al 87% de las entidades con un coste total del 13% del corpus. Aun incluyendo el coste de localizar las 5 semillas iniciales como la anotación de un documento completo por semilla, el coste aumentaría tan sólo en un punto porcentual, pasando a ser alrededor de un 7.2% y un 14% de los 485 documentos que contiene la colección.

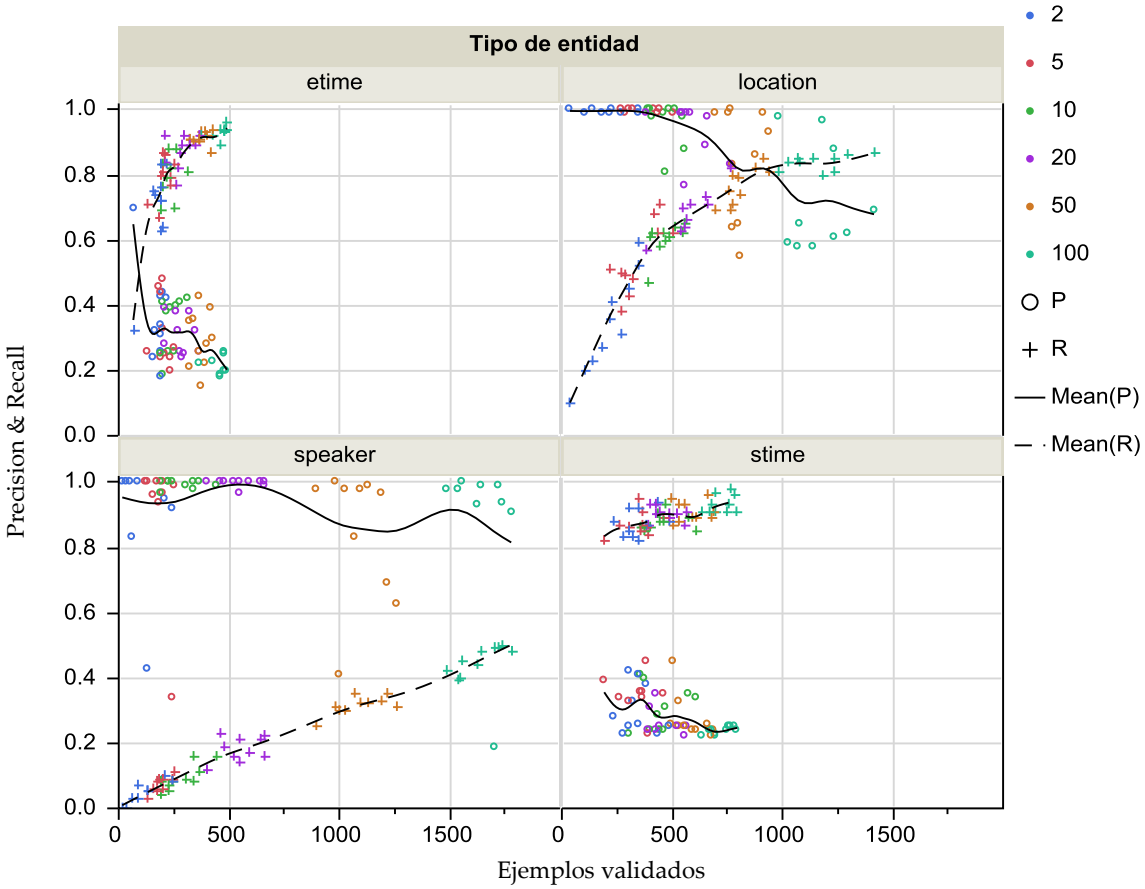


Figura 40. Tasas de precision y recall para los tipos de entidades del corpus seminarios.

Tabla 44. Relación precision , recall y costes de anotación (generación y validación) para cada tipo de entidad en el corpus seminarios tomando 5 semillas.

	P	R	Coste Generación (identificaciones)	Coste Validación (identificaciones)	Coste total aprox. (documentos)	Coste total aprox. (% corpus)
Etime	0.317	0.774	203	1150	30	6.2
Location	0.997	0.542	348	348	15	3.1
Speaker	0.92	0.071	184	65	6	1.2
Stime	0.341	0.872	343	2589	63	13

7.6.2. Jobs

Los resultados obtenidos de media para el corpus jobs pueden verse en la Tabla 45. Igual que para seminarios, vemos el coste de generación sumado al coste de validación de las identificaciones hechas por los patrones sobre el corpus. De nuevo los costes de anotación totales han sido aproximados en número de documentos y porcentaje del corpus que suponen, sabiendo que de media tenemos 312.786 tokens/documento.

Los resultados muestran que a partir de 5 semillas podemos anotar de media más de un 60% de las entidades, anotando el equivalente aproximado a tan sólo 8 documentos (un 2.7% del corpus). Y es posible llegar anotar más de un 80% de las entidades utilizando al menos 50 semillas iniciales y anotando el equivalente aproximado a entre 17 documentos, que supone menos del 6% del corpus.

Tabla 45. Relación Recall y coste de anotación medios para el corpus Jobs.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
Recall	**0.447**	**0.603**	**0.687**	**0.737**	**0.834**	**0.869**
Coste Generación (identificaciones)	110	189	260	340	492	617
Coste Generación (tokens)	381	712	983	1298	2038	2709
Coste Validación (identificaciones)	287	442	561	582	875	906
Coste Validación (tokens)	1083	1668	2118	2198	3304	3422
Coste total (tokens)	1464	2380	3101	3496	5342	6131
Coste total aprox. (documentos)	5	8	10	11	17	20
Coste total aprox. (% corpus)	**1.7**	**2.7**	**3.4**	**3.7**	**5.7**	**6.7**

Igual que en el caso de seminarios, si consideramos que el coste de identificar cada semilla inicial supone anotar un documento completo, tendríamos los costes de anotación mostrados en la Tabla 46. En este caso los costes totales, en porcentaje, son similares a los obtenidos de seminarios, pero el recall es bastante más elevado, con una media de reconocimiento de más del 80% de las entidades con alrededor del 20% del corpus.

Tabla 46. Relación Recall y coste de anotación medios, considerando que el coste de obtener una semilla equivale a la anotación de un documento completo.

	Semillas					
	2	**5**	**10**	**20**	**50**	**100**
Recall	0.447	0.603	0.687	0.737	0.834	0.869
Coste total aprox. (% corpus)	2.3	4.4	6.7	10.4	22	37.7

Un examen más exhaustivo por tipo de entidad (Tabla 47) muestra que con cinco semillas de partida, en 6 de las 18 entidades (*city, country, state, desired_years_experience, phone, post_date)* se observan tasas de recall superiores a 0.8, con una media de entre 3 y 8 documentos de coste de anotación. Esto supone anotar más de un 80% de esos tipos de entidad revisando menos de un 3% del corpus como máximo. Otros dos tipos de entidad, *salary* y *req_years_exp*, también presentan altas tasas de recall, superiores al 0.7 con similar coste. Son también destacables los tipos *id* y *req_degree,* con las que se consigue anotar más de un 50% de las entidades con tan sólo el equivalente a 7 y 2 documentos respectivamente.

Tabla 47. Tasas de precision y recall para cada tipo de entidad en el corpus jobs tomando 5 semillas. Se muestra el coste de anotación de generación de los patrones sumado al coste de validar las identificaciones, expresado todo ello en número de documentos aproximados a validar.

	P	R	Coste Generación (identificados)	Coste Validación (identificados)	Coste total aprox. (documentos)	Coste total aprox. (% corpus)
Application	0.636	0.336	266	667	11	3.7
Area	0.3	0.205	348	1320	25	8
City	0.962	0.879	250	579	8	2.7
Company	0.787	0.374	203	255	6	2
Country	0.982	0.882	109	307	5	1.7
Desired_degree	0.759	0.283	61	7	1	0.3
Desired_years_exp	0.684	0.835	168	55	3	1
Id	1	0.618	188	183	7	2.3
Language	0.637	0.399	289	1025	15	5
Phone	0.949	0.972	216	663	11	3.7
Platform	0.524	0.465	269	1034	15	5
Post_date	1	0.92	44	272	4	1.3
Recruiter	0.928	0.345	103	119	3	1
Req_degree	0.728	0.573	93	92	2	0.7
Req_years_exp.	0.38	0.739	217	439	8	2.7
Salary	0.958	0.776	122	110	2	0.7
State	0.939	0.824	146	400	6	2
Title	0.622	0.434	318	421	10	3.3

Lograr estas tasas en otras entidades requiere aumentar el número de semillas iniciales. Para casi todos los tipos de entidad al aumentar suficientemente el número de semillas iniciales se consiguen tasas de recall en torno al 0.8, con costes de entre 11 y 54 documentos. Las que exigen mayor esfuerzo de anotación son los tipos *application* y *area,* que alcanzan un recall del 0.72 y 0.62 respectivamente, con costes equivalentes a 36 y 52 documentos y 100 semillas iniciales. En estos casos, si consideramos que el coste de cada semilla equivale a revisar un documento completo, tendríamos que para llegar a reconocer un 72 y 62% de

las entidades respectivamente, tendríamos que anotar/validar ejemplos que equivaldrían a alrededor de la mitad del corpus.

7.7. Conclusiones

La metodología propuesta genera automáticamente patrones en forma de expresiones regulares o gramáticas IEG para el reconocimiento de entidades. Ha sido evaluada su capacidad de adaptación sobre 22 tipos de entidades muy variados repartidos en dos corpus diferentes.

Los resultados muestran que la generación de patrones IEG es competitiva con los resultados existentes en la literatura en términos de efectividad y costes de anotación al no disponer de corpus anotados previos. En el corpus seminarios los resultados son inferiores a otros trabajos al no contar en dos de las cuatro entidades con la información adicional procedente de la anotación al mismo tiempo de otros tipos de entidad. En cualquier caso nuevos experimentos permiten demostrar que este es realmente el problema, obteniéndose resultados competitivos al eliminar este sesgo. En el corpus jobs, con mayor número de tipos de entidad, se supera ampliamente la eficacia (un 62% más al usar tokens como unidad de anotación, y un 46% más al usar fragmentos de texto), y se reducen notablemente los costes de anotación incluso al compararlos con el uso del documento como unidad de anotación.

Se ha probado además la capacidad de la metodología para generar únicamente expresiones regulares en lugar de gramáticas IEG cuando el tipo de entidad responde a patrones exclusivamente de tipo carácter. La versatilidad de la metodología en este sentido resulta ser una ventaja adicional por la popularidad y facilidad de uso de este tipo de patrón. Ante esta situación se han obtenido elevadas tasas de efectividad a muy bajo coste de anotación, incluso utilizando semillas no pertenecientes al corpus. Estos datos muestran además la robustez del proceso ante la incorporación de semillas que no siguen exactamente los patrones de las entidades contenidas en el corpus.

La flexibilidad y potencia en el reconocimiento de esta metodología unida a la mantenibilidad de los patrones generados, y la ausencia de corpus anotados previos como requisito, la hacen portable ante nuevos corpus y entidades. Esta característica es muy útil también para la rápida creación de recursos anotados que favorezcan la evaluación en el área. En esta tarea los resultados muestran que con un coste de anotación equivalente a alrededor de un 20% del corpus (incluyendo que el coste de localizar una semilla pueda suponer la anotación completa de un documento), es posible anotar correctamente y sin falsos positivos más del 70% de las entidades (de media para cada tipo) en el corpus seminarios, y más del 80% en el corpus jobs.

Los resultados de los diferentes experimentos han mostrado además que a mayor número de semillas iniciales, mayor es la eficacia lograda, pero también los costes de anotación. El motivo es que con mayor variedad de semillas generalmente aumentan los potenciales patrones y por tanto las consultas al usuario. También es importante la adecuada selección de semillas. Esto puede observarse en los experimentos realizados partiendo de tan sólo dos semillas iniciales y alimentando con las resultantes un nuevo proceso. Se observa que a pesar de lograr mejorías en la efectividad, los costes de anotación pueden llegar a duplicarse o incluso triplicarse. Esto se debe a que el conjunto de semillas obtenido para el segundo proceso es menos representativo, por responder a patrones similares, que si se realizara una selección aleatoria de ese mismo número de semillas. Por este motivo la precision puede incluso llegar a aumentar, como ocurre en el corpus jobs, mientras que el recall disminuye en ambos corpus. Una importante ventaja en este sentido es la posibilidad que ofrece la metodología de usar semillas no pertenecientes al corpus usado, que abre la puerta al uso de bases de datos o listados para la generación de patrones que ya dispongan de ciertas garantías, evitando además la revisión del corpus para la localización de semillas.

Por último, aunque se ha obtenido una buena relación de efectividad/coste en los experimentos realizados, también se ha observado una importante variación entre los resultados obtenidos para unos y otros tipos de entidad. Por tanto, sería deseable ajustar los parámetros para cada uno de ellos. En concreto, los atributos a nivel token

seleccionados, el contexto y el umbral de similitud son factores que dependen del tipo de entidad. La adecuada selección de los atributos forma parte del área conocida como *feature engineering*, mientras que el contexto y el umbral de similitud podrían aproximarse mediante un análisis de las características de las entidades aportadas como semillas, lo que se plantea como trabajo futuro.

Capítulo 8
Conclusiones

El Reconocimiento de Entidades Nombradas es un área de investigación activa y con aplicación en muchas otras áreas relacionadas con la gestión de información, como Anotación Semántica, Sistemas de Búsqueda de Respuesta, Población de Ontologías y Análisis de Opinión. Pero la generación manual de patrones para el reconocimiento de entidades es compleja, por lo que es habitual el uso de métodos de generación automática. La efectividad de estos métodos está en muchos casos reñida con el esfuerzo que supone para el usuario anotar los corpus necesarios para el aprendizaje de estos patrones.

En este trabajo se diseña y evalúa un método de generación de patrones que no necesita corpus anotados previos. En su lugar parte de un pequeño conjunto de entidades, y durante el proceso de aprendizaje propone al usuario ciertas entidades para validar. Los resultados obtenidos muestran que reduce los costes necesarios para alcanzar los mismos niveles de efectividad de métodos similares.

Adicionalmente, en el método diseñado se han incorporado criterios de adaptabilidad que lo convierten en una solución práctica y de aplicación real ante diferentes tipos de entidades y dominios. Tradicionalmente no se le ha prestado la suficiente atención a esta cuestión porque los foros de evaluación internacionales se han limitado al reconocimiento de un conjunto muy limitado de entidades. Esto contrasta sin embargo con la variedad de tipos de entidad y dominios requeridos actualmente por usuarios y aplicaciones.

Los criterios de adaptabilidad considerados son la potencia y flexibilidad de los patrones, así como su grado de legibilidad y estandarización. Los patrones generados son capaces de reconocer el conjunto de los lenguajes regulares, e incluyen atributos de granularidad token y carácter. Su legibilidad facilita las modificaciones que deban realizarse ante pequeños cambios. La estandarización de su sintaxis contribuye a esta labor, además de favorecer la existencia de herramientas que faciliten su gestión. El modelo de representación de patrones diseñado, las gramáticas IEG-SRGS, y el método de generación capaz de generarlas de modo semi-automático, resulta ser en conjunto más adaptable que los métodos existentes según estos criterios. Además, al grado de adaptabilidad logrado se suma el uso de unidades de anotación no predefinidas. El método de generación de patrones es capaz de identificar y delimitar las entidades que el usuario deberá validar. De este modo se evita la dependencia de unidades predefinidas y se evitan costes de anotación innecesarios.

La orientación Web de las gramáticas IEG-SRGS supone una contribución adicional respecto a los métodos existentes. La sintaxis de las gramáticas está basada en lenguajes Web, lo que facilita su transferencia y almacenamiento en este medio. La posibilidad de identificarlos unívocamente facilita además la creación de repositorios distribuidos para la posterior reutilización de atributos y gramáticas completas o parte de ellas.

Otra de las ventajas es que el método de generación es capaz de generar no sólo gramáticas IEG-SRGS, sino también expresiones regulares. Las primeras además pueden ser transformadas a otros modelos, como los autómatas finitos en cascada. Se desliga así el método de generación de la necesidad de usar las gramáticas IEG-SRGS aquí diseñadas, aumentando su utilidad ante diversos escenarios.

En definitiva el trabajo realizado profundiza en el concepto de Entidad Nombrada y muestra que el Reconocimiento de Entidades Nombradas no sólo no puede considerarse un área resuelta, sino que necesita técnicas capaces de enfrentarse a la variedad de tipos de entidad y dominios que se le exige actualmente. Este dinamismo ha de ser integrado tanto en los foros de evaluación como en los métodos de generación de patrones. La contribución principal de este trabajo es presentar una alternativa ante los métodos de generación del

estado del arte que resulta ser más adaptable ante nuevos tipos y dominios, y más práctica al no requerir la anotación de corpus previos.

Capítulo 9
Trabajos futuros

Una de las limitaciones de este trabajo se refiere a los estudios de eficiencia. Aunque se ha analizado la complejidad en tiempo de reconocimiento de los patrones generados, no se ha analizado la complejidad que supone su generación, salvo en lo referido a algunos de los algoritmos que utiliza. Su estudio se plantea como trabajo futuro, así como las investigaciones destinadas a reducir el tiempo de reconocimiento de las gramáticas IEG. Concretamente, resultaría interesante investigar el uso de autómatas finitos deterministas cuando las gramáticas presentan la potencia de las expresiones regulares. También podrían realizarse estudios sobre la aplicabilidad de otros algoritmos de reconocimiento más eficientes en el caso de gramáticas IEG con la potencia de las gramáticas independientes del contexto. Estos estudios deberían complementarse además con comparaciones respecto a la complejidad en otros estándares, concretamente con las diferentes implementaciones del Common Pattern Specification Language (CPSL).

La transformación de las gramáticas IEG resultantes (o su generación directa) a modelos basados en CPSL, como el utilizado en el framework GATE, es otra de las propuestas. Dado su uso, y que no se conocen métodos capaces de generar directamente este tipo de patrones, resultaría de enorme interés profundizar en este tema.

En la misma línea de ampliar la aplicación del trabajo realizado, se propone la adaptación del método de generación de patrones a otros idiomas. Su aplicación a idiomas

basados en caracteres occidentales no parece presentar dificultad, pero sería necesario realizar un análisis en profundidad de esta cuestión para poder afirmarlo con seguridad.

Por otro lado, resultaría de interés profundizar en los diferentes algoritmos que integran el método de generación de patrones para localizar posibles mejoras, especialmente en el algoritmo para medir el grado de alineamiento y en el algoritmo de clustering utilizado. Del mismo modo, la opción de reconocer más de una entidad al mismo tiempo, ante casos donde nos interesa aprovechar posibles dependencias entre entidades, es una clara mejora a incorporar que puede reportar mejores resultados ante este tipo de escenarios. También la posibilidad de configurar automáticamente el método ante diferentes tipos de entidades puede ofrecer mejoras. Algunos de los parámetros configurables son la ventana de contexto, el umbral de similitud entre entidades o incluso los mejores atributos y el orden de aplicación. Esta información podría obtenerse de las características de las entidades iniciales aportadas.

Por último, durante la realización de este trabajo se ha observado la necesidad de profundizar en las medidas de esfuerzo del usuario en la anotación. Aunque el método diseñado ofrece al usuario potenciales entidades para validar, no se ha probado que esta opción requiera menos esfuerzo que anotar un texto equivalente de un documento. La investigación sobre el impacto en el usuario de los diferentes métodos de anotación es reciente, y aunque existen algunos trabajos al respecto, no son suficientes para asegurar que existen claras ventajas al validar en lugar de anotar texto. Por este motivo, aunque a priori pueda parecer que la validación ofrece menos dificultad, se ha considerado que ambas opciones son equivalentes en cuanto al coste de anotación. Ampliar los estudios sobre esta cuestión e identificar cuáles son las variables que influyen en la percepción de dificultad del usuario y su peso en la tarea de anotación contribuiría a diseñar métodos mejor adaptados a los diferentes perfiles de usuario, y a reducir el esfuerzo en la creación de los recursos necesarios para la evaluación en Reconocimiento de Entidades Nombradas.

Conclusions

Named Entity Recognition is an active field of research with application in many areas related with information management, such as Semantic Annotation, Question Answering, Ontology Population and Opinion Mining. However, manual generation of patterns for entity recognition is complex, so automatic methods are usually employed. The effectiveness of these methods largely depends on the effort on behalf of the user, who has to annotate the corpora necessary to automatically learn these patterns.

In this dissertation we design and evaluate a pattern learning method that does not need previously annotated corpora. Instead, it just requires a small set of entities, and throughout the learning process it proposes certain entities for the user to validate. Compared to similar methods in the state of the art, it reduces the cost required to achieve the same effectiveness rates.

Additionally, the designed method is adaptable to new types of entities and domains. This issue has not traditionally received much attention because international evaluation forums have been restricted to the recognition of a very limited set of entity types. But this contrasts with the variety of types of entities and domains currently required by users and applications.

Adaptability is achieved thanks to the power and flexibility of the generated patterns, named IEG-SRGS grammars, as well as their degree of readability and standardization. These grammars are capable of recognizing the set of regular languages, and they include features at the token and the character level. Their degree of readability and standardization facilitate the modifications that might be required. Standardization contributes further with existing tools that support their management. IEG-SRGS grammars, together with the method capable of generating them semi-automatically, are in conjunction more adaptable than existing methods under these criteria. Moreover, the learning method is capable of identifying and delimiting entities for the user to validate,

avoiding the dependency on predefined units and making the method more adaptable than others.

The Web orientation of the designed grammars is also an advantage with respect to existing models. The syntax facilitates their transfer through and storing in the Web. The possibility to uniquely identify them also helps in creating distributed repositories for the reuse of text features and grammars, both completely and partially in the case of the latter.

Another contribution is that the learning method is capable of generating not only IEG-SRGS grammars, but also typical regular expressions. Moreover, IEG-SRGS grammars can also be transformed to other models, such as finite state transducers. The method is thus independent of the necessity to use the IEG-SRGS grammars here designed, increasing its usefulness under diverse scenarios.

In summary, this dissertation researches the concept of Named Entity. It shows that Named Entity Recognition should not be considered a solved field of research and that it needs techniques capable to face the variety of entity and domain types currently required. This dynamism must be integrated in both, the evaluation forums and the pattern learning methods. The main contribution is the design of an alternative solution more practical in real applications because it does not require previously annotated corpora and it is more adaptable to new entity and domain types.

Future Work

One of the limitations of this work can be found in its efficiency analysis. Even though we analyzed the computational complexity in text recognition of the generated patterns, we did not analyze the full complexity of their generation except for some of the used algorithms. This study is proposed for future work, as well as research dedicated to reduce the time in text recognition. In particular, it would be interesting to analyze the use of deterministic finite automata when the generated grammars have the power of regular expressions. Also, further work can study the application of more efficient algorithms in text recognition when we are dealing with IEG grammars with the power of context free grammars. These studies should also be complemented with the analysis of the complexity when using other kinds of patterns, in particular the different implementations of the Common Pattern Specification Language (CPSL).

Another proposal is the transformation, or direct generation, of the resulting IEG grammars to CPSL-based models such as the one used in the GATE framework. We are not aware of methods capable of generating this kind of patterns automatically, so given the wide use of the framework this is a line with clear interest.

In the same line of widening the applicability of this work, we propose the adaptation of the pattern learning method to other languages. Its application to languages based on western alphabets has no apparent difficulty, but it is necessary to carry out a thorough analysis to confirm this with certainty.

On the other hand, it would be interesting to further improve the algorithms that make up the pattern learning method, especially with respect to the alignment and clustering algorithms. Likewise, the possibility to recognize more than one entity type at the same time, in cases where we can exploit potential dependencies among these types, is a clear line for improvement that may also provide better results in this type of scenarios. The possibility of automatically configuring the method for different types of entity may also improve results. Some of the configurable parameters are the context window, the

similarity threshold among entities or even the use of specific features or the order in which they are applied. These parameters could be set based on the characteristics of the initial seed entities.

Finally, during the development of this research we observed the need to dig into measures of user effort in the process of annotating or validating entities. Even though the designed method offers to the user potential entities to validate, we did not prove that this requires less effort than annotating an equivalent text in a document. Research on the impact that different annotation methods have on the user is quite recent, and even though there is some related literature, we cannot ensure there are clear advantages when validating instead of annotating. For this reason, and despite validating is intuitively easier, we considered that both alternatives are equivalent in terms of annotation cost. Further study of this question and the identification of the factors that influence the perceived difficulty of the task would contribute to the development of methods better adapted to different user profiles. It would also reduce the effort required for the creation of resources employed in the evaluation of Named Entity Recognition.

Apéndice. Ejemplo de funcionamiento del método de generación de patrones

Selección de semillas

Supongamos que queremos generar patrones para el reconocimiento de matrículas de coche. Disponemos de un corpus no anotado sobre el que localizamos algunos ejemplos de este tipo de entidad. Para ello cargamos el corpus en la herramienta implementada, que nos permite ver cada documento y señalar en él las entidades (ver Figura 41). En este caso localizamos las entidades "M125678" y "9875BGK".

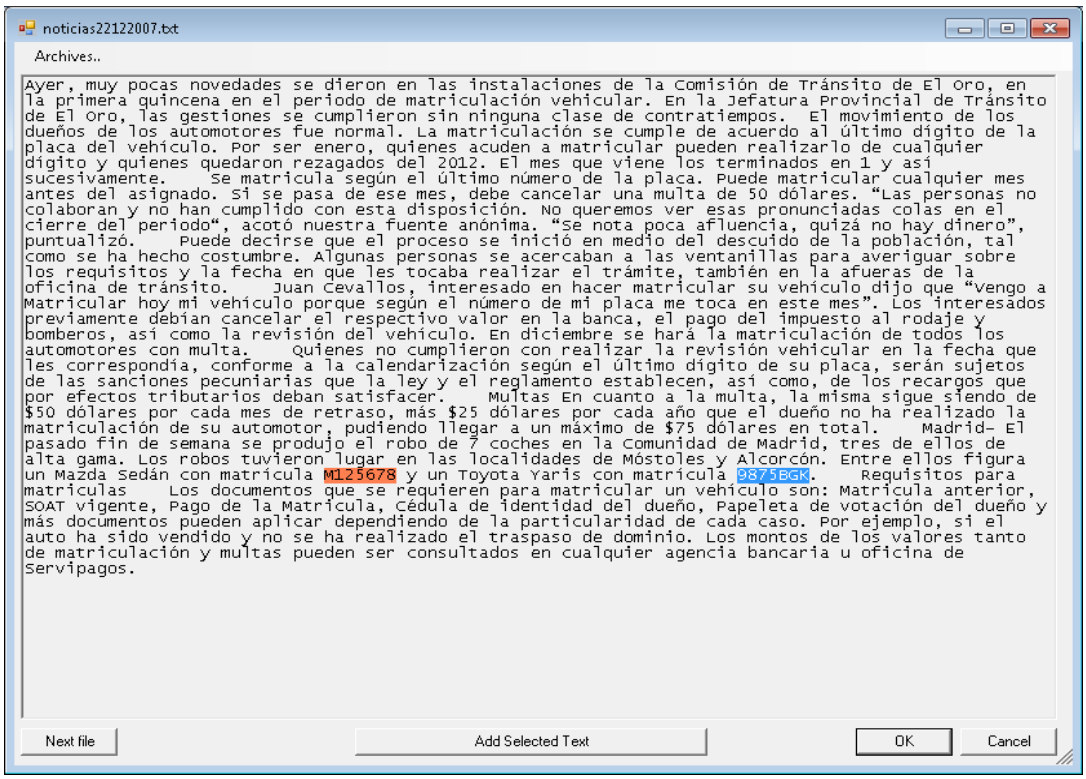


Figura 41. Selección de entidades iniciales en el corpus no anotado. Se revisan los documentos y al localizar una entidad se presiona "Add Selected Text". Estas entidades (que aparecerán marcadas en naranja) serán las semillas para iniciar el proceso de aprendizaje de patrones.

Fase I: Atributos de granularidad carácter

En primer lugar se añade el contexto seleccionado a cada entidad. En esta fase, se añadirán únicamente los caracteres delimitadores.

Atributo carácter

Se aplica el algoritmo de clustering sobre las semillas. Sobre cada uno de los clusters resultantes con más de una entidad se aplicará el algoritmo de identificación de patrones. En este caso, ambas entidades son agrupadas en el mismo clúster.

El algoritmo de identificación de patrones localiza entonces los patrones posicionales mejor alineados entre los caracteres de la entidad y de su contexto de forma independiente. Los patrones resultantes forman las reglas candidatas, que determinarán qué caracteres han de aparecer siempre en las entidades y en qué posición. En este caso las reglas candidatas son las que vemos sombreadas en la Figura 42.

␣	M	1	2	5	6	7	8	␣
␣	9	8	7	5	B	G	K	.

Figura 42. Reglas candidatas resultantes (en naranja y amarillo). Aparece separado el contexto de la entidad.

A continuación se aplica el proceso de aprendizaje activo. Con el objetivo de validar las reglas candidatas, se localizarán fragmentos en el texto que no encajen con al menos una de las reglas candidatas pero sí con la representación Unicode (el siguiente atributo a analizar) de los restantes caracteres. Con el objetivo de localizar entidades similares, se localizarán además fragmentos de texto similares a la representación Unicode de las entidades. Los fragmentos de texto localizados, limitados en número según la configuración, serán mostrados al usurio para que los valide (ver Figura 43). Aquellas que resultan ser realmente entidades son incorporadas al conjunto de semillas.

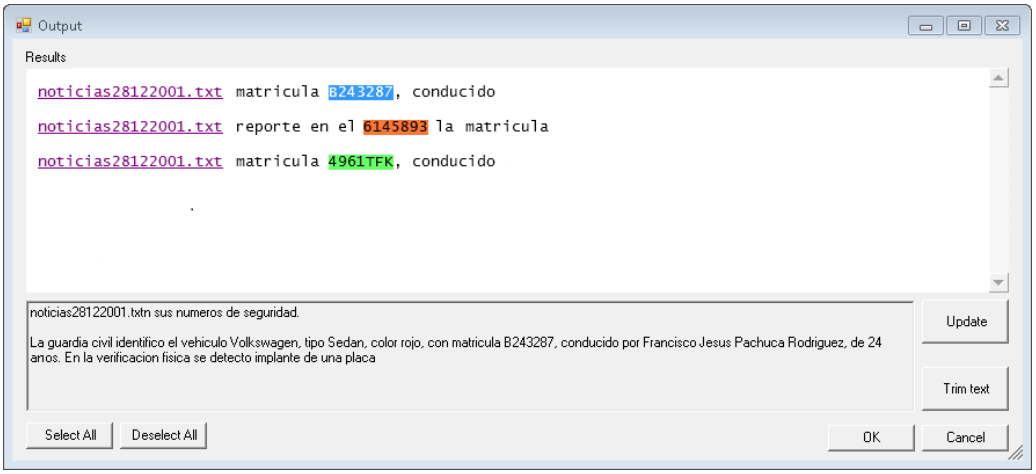


Figura 43. Proceso de validación de potenciales entidades localizadas por el sistema en el corpus no anotado. Las entidades aparecen señaladas en verde. El usuario pude pulsar sobre ellas para indicar que no son válidas (aparecen entonces marcadas en rojo). También es posible modificar los límites de cada entidad en el cuadro de edición inferior.

Tras este proceso son incorporadas dos nuevas entidades, con su contexto correspondiente, y se aplica de nuevo el algoritmo de identificación de patrones sobre ellas. En este caso el mejor patrón posicional corresponde al espacio delimitador (ver Figura 44). Dado que esta regla candidata ya había sido probada previamente, pasa a convertirse en regla válida (de lo contrario, se aplicaría el proceso de aprendizaje activo de nuevo).

␣	M	1	2	5	6	7	8
␣	9	8	7	5	B	G	K
␣	B	2	4	3	2	8	7
␣	4	9	6	1	T	F	K

␣
.
,
,

Figura 44. Nuevas reglas candidatas resultantes (en amarillo) de los patrones posicionales mejor alineados tanto en el contexto como en la propia entidad.

Se aplica de nuevo el algoritmo de identificación de patrones sobre los restantes caracteres de las entidades y no se encuentran más patrones posicionales. Finaliza el proceso.

Atributo Unicode

Los caracteres de las entidades que no han formado regla son transformados a su tipo Unicode correspondiente. Se aplica entonces el algoritmo de clustering sobre ellas. Como

resultado en este caso se genera un único cluster que agrupa a todas las entidades. Posteriormente se aplica sobre el cluster el algoritmo de identificación de patrones.

M125678	␣	M/m	Nd	Nd	Nd	Nd	Nd	Nd	Zs
B243287	␣	B/b	Nd	Nd	Nd	Nd	Nd	Nd	Po
9875BGK	␣	Nd	Nd	Nd	Nd	B/b	G/g	K/k	Po
4961TFK	␣	Nd	Nd	Nd	Nd	T/t	F/f	K/k	Po

Figura 45. Reglas candidatas resultantes (en naranja, amarillo, azul y rojo). En gris se muestran las reglas creadas en fases previas.

Se excluyen de la búsqueda de patrones aquellos elementos que ya han formado regla previamente. Sobre los restantes, se identifican los mejores, que formarán las reglas candidatas. En la Figura 45 se observa que son cuatro las reglas candidatas. A pesar de no estar los elementos que las forman alineados verticalmente, el grado de alineación entre ellas es máximo. De este modo, en lugar de las tres posibles reglas formadas por la alineación vertical de los elementos de tipo "Nd", se seleccionan cuatro candidatas que también tienen máximo grado de alineación según el algoritmo utilizado.

Se aplica entonces el proceso de aprendizaje activo. Para ello los caracteres que no pertenecen a las reglas candidatas o a las creadas en fases previas, son transformados a su respectiva agrupación Unicode (siguiente atributo a analizar).

En este caso no se encuentran nuevas entidades. Se aplica de nuevo el algoritmo de identificación de patrones, y dado que el conjunto de entidades es el mismo, las reglas candidatas resultantes también lo son. Puesto que las reglas candidatas son las mismas (ya han sido validadas con el proceso de aprendizaje activo) pasan a convertirse en reglas válidas.

El proceso finaliza al no encontrarse más patrones posicionales entre los elementos que no han formado regla previamente.

Atributo Grupo Unicode

Como en los casos anteriores, los elementos que no han formado regla se transforman en su correspondiente valor según la agrupación de tipos Unicode establecida. Se aplica el

algoritmo de clustering y en este caso resultan dos clusters: uno para las matrículas formadas por una letra seguida de seis dígitos (cluster 1), y otro para las entidades formadas por cuatro dígitos y tres letras (cluster 2).

Cluster 1

Se aplica el algoritmo de identificación de patrones sobre las entidades del cluster. Se localizan tres reglas candidatas (ver Figura 46). No se localizan en el corpus nuevas entidades que las invaliden, por lo que estas reglas pasan a ser válidas. Finaliza el proceso al no encontrarse más patrones.

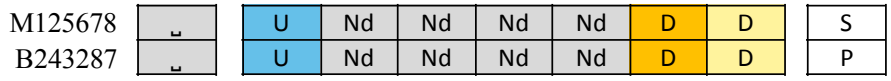


Figura 46. Reglas candidatas resultantes (enazul, naranja y amarillo). En gris se muestran las reglas creadas en fases previas.

Cluster 2

Se sigue el mismo proceso que para el cluster 1 y finalmente son creadas las cuatro reglas mostradas en la Figura 47.

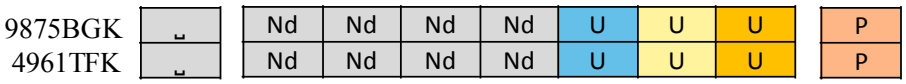


Figura 47. Reglas candidatas resultantes (en azul, amarillo, naranja y rojo). En gris se muestran las reglas creadas en fases previas.

Fase II. Atributos de granularidad token

En la segunda fase de este algoritmo es posible aplicar cualquier atributo de granularidad token que se considere conveniente. En este caso aplicaremos el atributo *string*, que devolverá el texto de cada token[28].

[28] Este atributo puede resultar redundante, pero como veremos, es útil para el análisis del contexto, donde no hemos considerado conveniente aplicar atributos de granularidad carácter por el riesgo de crear reglas demasiado específicas (*overfitting).*

En primer lugar, se agrupan los caracteres de las entidades en tokens y se añade el contexto predefinido a cada entidad (en este caso se tomarán dos tokens en cada extremo). A continuación se procesan de modo independiente cada uno de los atributos.

Atributo *string*

Se transforma cada token al valor correspondiente del atributo. A partir de aquí, el proceso es el mismo que para cualquiera de los atributos carácter: se aplica el mismo algoritmo de clustering (en este caso resultan dos clusters) y sobre cada cluster se aplica el mismo algoritmo de identificación de patrones.

Cluster 1

Se localizan los mejores patrones sobre los valores de los atributos, de nuevo de forma separada para el contexto y la propia entidad. En este caso resultan tres reglas, todas del contexto. Las reglas candidatas son contrastadas con el proceso de aprendizaje activo, localizando fragmentos de texto que coinciden con los valores del atributos en cada entidad y su contexto.

con	matricula	B243287	conducido	por
blanco	matricula	4961TFK	conducido	por

Figura 48. Reglas candidatas y finalmente creadas (en azul, naranja y amarillo).

En este caso no son localizadas nuevas entidades, por lo que las reglas candidatas pasan a ser válidas. Finaliza el proceso al no encontrarse nuevos patrones entre los elementos que no han formado regla.

Cluster 2

con	matricula	M125678	y	un
con	matricula	9875BGK	Requisitos	para

Figura 49. Reglas candidatas y finalmente creadas (en amarillo y naranja).

Se sigue el mismo proceso que para el cluster 1 y finalmente son creadas las dos reglas mostradas en la Figura 49.

Gramática IEG resultante

Como resultado de la primera fase se fijarán determinados caracteres, tipos Unicode o agrupaciones Unicode en las entidades y su contexto (caracteres delimitadores en este caso). Como resultado de la segunda fase, se fijarán ciertos valores de atributos a los tokens de las entidades o su contexto (dos tokens en cada extremo en este caso).

El patrón resultante, modelado como una gramática IEG, puede verse en el Código 28. Cada regla creada en la primera fase genera un no terminal cuyo cuerpo de producción contiene el símbolo que la ha generado y al que sustituye. Cada regla creada en la segunda fase es convertida en una función asociada al terminal que representa al token sobre el que se crea. Estos tokens pueden ser del contexto o del propio texto de las entidades (A y B).

```
S→  T1 R T2 R1 A
S→  T1 R T2 B R2
S→  T2 R1 A  T3 R T4
S→  T2 R B R2 T3 R T4
A→  U Nd{4} D{2}
B→  Nd{4} U{3}
R1→
R2→ P
T1→ T
T2→ T
T3→ T
T4→ T
T→  [token]
R→  [carácter no alfanumérico]+
FT1 = {(string,"con")}
FT2 = {(string,"matricula")}
FT3 = {(string,"conducido")}
FT4 = {(string,"por")}
```

Código 28. Gramática IEG resultante del método de generación de patrones. Por simplicidad, las reglas generadas en el cuerpo de los no terminales A y B han sido omitidas y en los cuerpos de producción se utilizan directamente los símbolos utilizados en los ejemplos y otras simplificaciones entre corchetes.

Esta gramática podría ser posteriormente optimizada, puesto que es habitual que conjuntos de entidades respondan a patrones iguales o muy similares.

Bibliografía

AGIRRE, E., ANSA, O., HOVY, E., & MARTÍNEZ, D., "Enriching very large ontologies using the WWW", *Ontology learning workshop*, 2000.

ALFONSECA, E. & MANANDHAR, S., "An unsupervised method for general named entity recognition and automated concept discovery", *International Conference on General WordNet*, 2002.

APPELT, D.E. & ONYSHKEVYCH, B., "The common pattern specification language", *TIPSTER Text Program: Phase III*, pp. 23–30, 1998.

ARONSON, A.R., "Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program", *American Medical Informatics Association Annual Symposium*, pp. 17–21, 2001.

ARTILES, J., GONZALO, J., & SEKINE, S., "The semeval-2007 WePS evaluation: Establishing a benchmark for the web people search task", *Semeval*, pp. 64–69, 2007.

ASAHARA, M. & MATSUMOTO, Y., "Japanese named entity extraction with redundant morphological analysis", *Human Language Technology Conference*, pp. 8–15, 2003.

BALOG, K., SERDYUKOV, P., & VRIES, A.P. DE, "Overview of the TREC 2010 Entity Track", *The Nineteenth Text REtrieval Conference*, 2010.

BIKEL, D.M., MILLER, S., SCHWARTZ, R., & WEISCHEDEL, R., "Nymble: a high-performance learning name-finder", *Conference on Applied Natural Language Processing*, pp. 194–201, 1997.

BIZER, C., LEHMANN, J., KOBILAROV, G., AUER, S., BECKER, C., CYGANIAK, R., & HELLMANN, S., "DBpedia - A crystallization point for the Web of Data", *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, n.3, pp. 154–165, 2009a.

BIZER, C., HEATH, T., & BERNERS-LEE, T., "Linked Data: the story so far", *International Journal on Semantic Web and Information Systems*, vol. 5, n.3, pp. 1–22, 2009b.

BOGURAEV, B.K., "Annotation-based finite state processing in a large-scale NLP architecture", in *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003*, John Benjamins B.V., pp. 61–77, 2004.

BONTCHEVA, K., DIMITROV, M., MAYNARD, D., TABLAN, V., & CUNNINGHAM, H., "Shallow methods for named entity coreference resolution", *Chaines de references et resolveurs d'anaphores workshop (Conférence sur le Traitement Automatique des Langues Naturelles)*, 2002.

BORREGA, O., TAULÉ, M., & MARTI, M.A., "What do we mean when we speak about Named Entities", *Corpus Linguistics Conference*, 2007.

BORTHWICK, A., STERLING, J., AGICHTEIN, E., & GRISHMAN, R., "Exploiting diverse knowledge sources via maximum entropy in named entity recognition", *Workshop on Very Large Corpora (Annual Meeting of the Association for Computational Linguistics and Conference on Computational Linguistics)*, pp. 152–160, 1998.

BRAUER, F., RIEGER, R., MOCAN, A., & BARCZYNSKI, W.M., "Enabling Information Extraction by Inference of Regular Expressions from Sample Entities", *Conference on Information and Knowledge Management*, pp. 1285–1294, 2011.

BRILL, E. & RESNIK, P., "A Rule-Based Approach to Prepositional Phrase Attachment Disambiguation", *Conference on Computational Linguistics*, pp. 1198–1204, 1994.

BRIN, S., "Extracting Patterns and Relations from the World Wide Web", *Selected papers from the International Workshop on The World Wide Web and Databases*, pp. 172–183, 1998.

BRUNSTEIN, A., "Annotation guidelines for answer types (ref. LDC2005T33)", *Linguistic Data Consortium*, 2002, disponible en: http://www.ldc.upenn.edu/Catalog/docs/LDC2005T33/BBN-Types-Subtypes.html.

CALIFF, M.E. & MOONEY, R.J., "Bottom-up relational learning of pattern matching rules for information extraction", *Journal of Machine Learning Research*, vol. 4, n.2, pp. 177–210, 2004.

CHEN, H., "Machine learning for information retrieval: neural networks , symbolic learning , and genetic algorithms", *Journal of the American Society for Information Science*, vol. 46, n.3, pp. 194–216, 1995.

CHINCHOR, N., ROBINSON, P., & BROWN, E., "HUB-4 named entity task definition (v4.8)", *National Institute of Standards and Technology*, 1998.

CHINCHOR, N. & ROBINSON, P., "MUC-7 named entity task definition", *Conference on Message Understanding*, 1997.

CHITICARIU, L., LI, Y., RAGHAVAN, S., & REISS, F.R., "Enterprise information extraction: recent developments and open challenges", *SIGMOD International Conference on Management of Data*, pp. 1257–1258, 2010a.

CHITICARIU, L., REISS, F.R., & VAITHYANATHAN, S., "SystemT: an algebraic approach to declarative information extraction", *Annual Meeting of the Association for Computational Linguistics*, pp. 128–137, 2010b.

CIARAMITA, M. & ALTUN, Y., "Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger", *Conference on Empirical Methods in Natural Language Processing*, pp. 594–602, 2006.

CIRAVEGNA, F., COURT, R., STREET, P., & SHEFFIELD, S., "(LP)2 , an adaptive algorithm for information extraction from web-related texts types of induced rules", *Workshop on Adaptive Text Extraction and Mining (International Joint Conferences on Artificial Intelligence)*, 2001.

CIRAVEGNA, F. & WILKS, Y., "Designing adaptive information extraction for the Semantic Web in Amilcare", in *Annotation for the Semantic Web (Frontiers in Artificial Intelligence and Applications series)*, IOS Press, pp. 112–127, 2003a.

CIRAVEGNA, F. & WILKS, Y., "Designing adaptive information extraction for the semantic web in amilcare", IOS Press, pp. 112–127, 2003b.

COLLINS, M. & SINGER, Y., "Unsupervised models for named entity classification", *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 189–196, 1999.

COOK, T.D. & CAMPBELL, D.T., "Quasi-experimentation: design and analysis issues for field settings", Houghton Mifflin Company, 1979.

CRESTAN, E. & LOUPY, C. DE, "Natural language processing for browse help", *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 488–489, 2004.

CUCCHIARELLI, A. & VELARDI, P., "Unsupervised named entity recognition using syntactic and semantic contextual evidence", *Computational Linguistics*, vol. 27, n.1, pp. 123–131, 2001.

CUCERZAN, S., "Large-scale named entity disambiguation based on Wikipedia data", *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 708–716, 2007.

CUNNINGHAM, H., GAIZAUSKAS, R., WAKAO, T., HUMPHREYS, K., & WILKS, Y., "Description of the LaSIE system as used for MUC-6", *Message Understanding Conference*, pp. 207–220, 1995.

CUNNINGHAM, H., "Information extraction, automatic", in *Encyclopedia of Language and Linguistics*, Elsevier, pp. 665–677, 2005.

CUNNINGHAM, H., MAYNARD, D., BONTCHEVA, K., TABLAN, V., ASWANI, N., ROBERTS, I., GORRELL, G., FUNK, A., ROBERTS, A., DAMLJANOVIC, D., HEITZ, T., GREENWOOD, M.A., SAGGION, H., PETRAK, J., LI, Y., PETERS, W., & ET AL., "Developing language processing components with version 7 (a user guide)", *Department of Computer Science, The University of Sheffield*, 2013.

CURRAN, J.R. & CLARK, S., "Language independent NER using a maximum entropy tagger", *Conference on Natural Language Learning*, pp. 164–167, 2003.

DEMARTINI, G., IOFCIU, T., & VRIES, A. DE, "Overview of the INEX 2009 Entity Ranking Track", in *INEX 2009 Workshop Pre-proceedings*, IR Publications, pp. 233–237, 2009.

DOAN, A., RAMAKRISHNAN, R., & VAITHYANATHAN, S., "Managing Information Extraction: state of the art and research directions", *ACM SIGMOD International Conference on Management of Data*, pp. 799–800, 2006.

DODDINGTON, G., MITCHELL, A., PRZYBOCKI, M., RAMSHAW, L., STRASSEL, S., & WEISCHEDEL, R., "The Automatic Content Extraction (ACE) program: tasks, data, and evaluation", *International Conference on Language Resources and Evaluation*, vol. 4, pp. 837–840, 2004.

DROZDZYNSKI, W., KRIEGER, H.-U., PISKORSKI, J., SCHÄFER, U., & XU, F., "Shallow processing with unification and typed feature structures: foundations and applications", *Künstliche Intelligenz*, vol. 1, pp. 17–23, 2004.

ETZIONI, O., CAFARELLA, M., DOWNEY, D., POPESCU, A.M., SHAKED, T., SODERLAND, S., WELD, D.S., & YATES, A., "Unsupervised named-entity extraction from the Web: An experimental study", *Artificial intelligence.*, vol. 165, n.1, pp. 91–134, 2005.

FERRÁNDEZ, O., KOZAREVA, Z., MONTOYO, A., & MUNOZ, R., "Nerua: sistema de deteccion y clasificacion de entidades utilizando aprendizaje automatico", *Procesamiento del Lenguaje Natural*, vol. 35, pp. 37–44, 2005.

FISHER, D., SODERLAND, S., MCCARTHY, J., FENG, F., & LEHNER, W., "Description of the UMASS system as used for MUC- 6", *Conference on Message Understanding*, pp. 127–140, 1995.

FREITAG, D., "Toward general-purpose learning for information extraction retargetability", *International Conference on Computational Linguistics*, vol. 1, pp. 404–408, 1998.

FREITAG, D. & KUSHMERICK, N., "Boosted wrapper induction", *National Conference on Artificial Intelligence*, vol. 145, pp. 577–583, 2000.

GAIZAUSKAS, R., DEMETRIOU, G., ARTYMIUK, P., & WILLET, P., "Protein structures and information extraction from biological texts: the Pasta system", *Bioinformatics*, vol. 19, n.1, pp. 135–143, 2003.

GANTZ, J. & REINSEL, D., "The Digital Universe Decade, Are You Ready?", *International Data Corporation (IDC)*, 2010, disponible en: http://www.emc.com/digital_universe.

GATE PROJECT TEAM, "GATE information extraction example", *GATE: General Architecture for Text Engineering*, http://gate.ac.uk/ie/ie_example.html, accedido por última vez: February 21, 2013.

GRISHMAN, R. & SUNDHEIM, B., "Message Understanding Conference-6: a brief history", *Conference on Computational Linguistics*, pp. 466–471, 1996.

HACHEY, B., ALEX, B., & BECKER, M., "Investigating the effects of selective sampling on the annotation task", *Conference on Natural Language Learning*, pp. 144–151, 2005.

HANDSCHUH, S. & STAAB, S., "Authoring and annotation of web pages in CREAM", *International World Wide Web Conference*, pp. 462–473, 2002.

HARMAN, D.K., "Information retrieval evaluation", *Synthesis Lectures on Information Concepts, Retrieval and Services*, vol. 3, n.2, pp. 1–119, 2011.

HAYES, P.J. & WEINSTEIN, S.P., "CONSTRUE/TIS: a system for content-based indexing of a database of news stories", *Annual Conference on Innovative Applications of Artificial Intelligence*, vol. 97, pp. 49–64, 1990.

HEARST, M.A., "Automatic acquisition of hyponyms from large text corpora", *International Conference on Computational Linguistics*, 1992.

HOGUE, A. & KARGER, D., "Thresher: automating the unwrapping of semantic content from the World Wide Web", *International Conference on World Wide Web*, pp. 86–95, 2005.

HOPCROFT, J.E., RAJEEV, M., & ULLMAN, J.D., "Introduction to automata theory, languages and computation", (3rd ed.), Addisson-Wesley, 2006.

HUNT, A. & MCGLASHAN, S., "Speech Recognition Grammar Specification Version 1.0", *World Wide Web Consortium (W3C)*, 2004, disponible en: http://www.w3.org/TR/2004/REC-speech-grammar-20040316/.

JACOBS, P.S. & RAU, L.F., "SCISOR: extracting information from on-line news", *Communications of the ACM*, vol. 33, n.11, pp. 88–97, 1990.

JI, H. & GRISHMAN, R., "Data selection in semi-supervised learning for name tagging", *Workshop on Information Extraction Beyond the Document (International Conference on Computational Linguistics)*, pp. 48–55, 2006.

JONES, R., "Learning to extract entities from labelled and unlabelled text", Ph.D., *School of Computer Science, Carnegie Mellon University*, 2005.

JONQUET, C., MUSEN, M.A., & SHAH, N., "A system for ontology-based annotation of biomedical data", *Lecture Notes in Computer Science (Conference on Data Integration in the Life Sciences)*, vol. 5109, pp. 144–152, 2008.

KAHAN, J., KOIVUNEN, M.J., PRUD'HOMMEAUX, E., & SWICK, R., "Annotea: an open RDF infrastructure for shared web annotations", *Computer Networks*, vol. 39, n.5, pp. 589–608, 2002.

KATZER, J., COOK, K.H., & CROUCH, W.W., "Evaluating information: a guide for users of social science research", (4th ed.), McGraw-Hill, 1998.

KAZAMA, J. & TORISAWA, K., "Exploiting Wikipedia as external knowledge for named entity recognition", *Joint Conference on Empirical Methods in Natural Language Processing and Conference on Natural Language Learning*, pp. 698–707, 2007.

KIM, JD., OHTA, T., TETEISI, Y., & TSUJII, J., "GENIA corpus: a semantically annotated corpus for bio-textmining", *Bioinformatics*, vol. 19 (suppl., pp. i180–i182, 2003.

KOZAREVA, Z., BONEV, B., & MONTOYO, A., "Self-training and co-training applied to spanish named entity recognition", *Lecture Notes in Computer Science (MICAI 2005: Advances in Artificial Intelligence)*, vol. 3789, pp. 770–779, 2005.

KRIPKE, S.A., "Naming and Necessity", Harvard University Press, 1980.

KUSHMERICK, N., "Wrapper induction for information extraction", Ph.D., *University of Washington*, 1997.

LABSKÝ, M., SVÁTEK, V., NEKVASIL, M., & RAK, D., "The Ex Project: web information extraction using extraction ontologies", *Lecture Notes in Computer Science (Knowledge Discovery Enhanced with Semantic and Social Information)*, vol. 220, pp. 71–88, 2009.

LAPORTE, J., "Rigid Designators", in *The Stanford Encyclopedia of Philosophy*, The Metaphysics Research Lab, Center for the Study of Language and Information, Stanford University, 2011.

LAWS, F. & SCHÜTZE, H., "Stopping criteria for active learning of named entity recognition", *International Conference on Computational Linguistics*, vol. 1, pp. 465–472, 2008.

LEVENSHTEIN, V.I., "Binary codes capable of correcting deletions, insertions and reversals", *Soviet Physics Doklady*, vol. 10, n.8, pp. 707–710, 1966.

LI, X., MORIE, P., & ROTH, D., "Identification and tracing of ambiguous names: discriminative and generative approaches", *National Conference on Artificial Intelligence*, pp. 419–424, 2004.

LI, Y., BONTCHEVA, K., & CUNNINGHAM, H., "Adapting SVM for data sparseness and imbalance: a case study in information extraction", *Natural Language Engineering*, vol. 15, n.2, pp. 241–271, 2008a.

LI, Y., KRISHNAMURTHY, R., RAGHAVAN, S., VAITHYANATHAN, S., & JAGADISH, H.V., "Regular expression learning for information extraction", *Conference on Empirical Methods in Natural Language Processing*, pp. 21–30, 2008b.

LIN, D. & PANTEL, P., "Induction of semantic classes from natural language text", *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 317–322, 2001.

MAEDCHE, A. & STAAB, S., "Ontology learning for the Semantic Web", *IEEE Intelligent Systems*, vol. 16, n.2, pp. 72–79, 2001.

MALOUF, R., "Markov models for language-independent named entity recognition", *Conference on Natural Language Learning*, pp. 187–190, 2002.

MANI, I. & WILSON, G., "Robust temporal processing of news", *Annual Meeting of the Association for Computational Linguistics*, vol. 38, pp. 69–76, 2000.

MANNING, C.D., RAGHAVAN, P., & SCHÜTZE, H., "Introduction to information retrieval", Cambridge University Press, 2008.

MARRERO, M., SÁNCHEZ-CUADRADO, S., MORATO, J., & ANDREADAKIS, Y., "Evaluation of named entity extraction systems", *International Conference on Intelligent Text Processing and Computational Linguistics*, 2009a.

MARRERO, M., SÁNCHEZ-CUADRADO, S., MORATO, J., & ANDREADAKIS, Y., "Evaluation of named entity extraction systems", *Research in Computing Science (Special Issue: Advances in Computational Linguistics)*, vol. 41, pp. 47–58, 2009b.

MARRERO, M., URBANO, J., MORATO, J., & SÁNCHEZ-CUADRADO, S., "On the definition of patterns for semantic annotation", *CIKM Workshop on Exploiting Semantic Annotations in Information Retrieval*, pp. 15–16, 2010a.

MARRERO, M., SÁNCHEZ-CUADRADO, S., URBANO, J., MORATO, J., & MOREIRO, J.A., "Tratamiento léxico en los sistemas de recuperación y representación de información en biomedicina", *El Profesional de la Información*, vol. 19, n.3, pp. 246–254, 2010b.

MARRERO, M., SANCHEZ-CUADRADO, S., URBANO, J., MORATO, J., & MOREIRO, J.A., "Information retrieval systems adapted to the biomedical domain", *ACM Computing Research Repository*, 2012a, disponible en: http://arxiv.org/abs/1203.6845.

MARRERO, M., URBANO, J., SÁNCHEZ-CUADRADO, S., MORATO, J., & GÓMEZ-BERBÍS, J.M., "Named Entity Recognition: fallacies, challenges and opportunities", *Computer Standards & Interfaces (to appear)*, 2012b.

MCCARTHY, J. & LEHNERT, W.G., "Using decision trees for coreference resolution", *International Joint Conferences on Artificial Intelligence*, pp. 1050–1055, 1995.

MIKHEEV, A., "A knowledge-free method for capitalized word disambiguation", *Annual Meeting of the Association for Computational Linguistics*, pp. 159–166, 1999.

MITCHELL, M.L. & JOLLEY, J.M., "Research design explained", (7th ed.), Wadsworth Publishing, 2009.

MOENS, M.F., "Information extraction: algorithms and prospects in a retrieval context", Springer, 2006.

MOLLA, D., ZAANEN, M. VAN, & SMITH, D., "Named entity recognition for question answering", *Australasian Language Technology Workshop*, pp. 51–58, 2006.

MONTGOMERY, D.C., "Design and analysis of experiments", (7th ed.), John Wiley And Sons, 2009.

MUC-6 PROGRAM COMMITTEE, "Named Entity Task Definition, version 2.0", 1995, disponible en: http://cs.nyu.edu/cs/faculty/grishman/NEtask20.book_1.html.

NADEAU, D., TURNEY, P., & MATWIN, S., "Unsupervised named entity recognition: generating gazetteers and resolving ambiguity", *Lecture Notes in Computer Science (Advances in Artificial Intelligence)*, vol. 4013, pp. 266–277, 2006.

NADEAU, D., "Semi-supervised named entity recognition: learning to recognize 100 entity types with little supervision", Ph.D., *School of Information Technology and Engineering, University of Ottawa*, 2007.

NADEAU, D. & SEKINE, S., "A survey of named entity recognition and classification", *Linguisticae Investigationes*, vol. 30, n.7, 2007.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST), "Automatic Content Extraction Evaluation (ACE08). Official Results.", *National Institute of Standards and Technology*, 2008, disponible en: http://www.itl.nist.gov/iad/mig/tests/ace/2008/doc/ace08_eval_official_results_20080929.html.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST), "Task description for Knowledge Base Population at TAC 2009", 2009, disponible en: http://www.nist.gov/tac/.

NG, V. & CARDIE, C., "Improving machine learning approaches to coreference resolution", *Annual Meeting of the Association for Computational Linguistics*, pp. 104–111, 2002.

NOTHMAN, J., MURPHY, T., & CURRAN, J., "Analysing Wikipedia and gold-standard corpora for NER training", *Conference of the European Chapter of the Association for Computational Linguistics*, pp. 612–620, 2009.

NOY, N.F. & MCGUINESS, D.L., "Ontology development 101: a guide to creating your first ontology", 2001, disponible en: http://bmir.stanford.edu/publications.

ORALLO, J.H., QUINTANA, M.J.R., & RAMÍREZ, C.F., "Introducción a la minería de datos", Pearson Educación, 2005.

PALMER, D.D. & DAY, D.S., "A statistical profile of the named entity task", *Conference on Applied Natural Language Processing*, pp. 190–193, 1997.

PASCA, M., "Weakly-supervised discovery of named entities using web search queries", *ACM Conference on Information and Knowledge Management*, pp. 683–690, 2007.

PETASIS, G., CUCCHIARELLI, A., VELARDI, P., PALIOURAS, G., KARKALETSIS, V., & SPYROPOULOS, C.D., "Automatic adaptation of proper noun dictionaries through cooperation of machine learning and probabilistic methods", *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 128–135, 2000.

PETASIS, G., VICHOT, F., WOLINSKI, F., PALIOURAS, G., KARKALETSIS, V., & SPYROPOULOS, C.D., "Using machine learning to maintain rule-based named-entity recognition and classification systems", *Annual Meeting on Association for Computational Linguistics*, pp. 426–433, 2001.

PIANTA, E., GIRARDI, C., & ZANOLI, R., "The TextPro tool suite", *International Conference on Lenguage Resources and Evaluation*, 2008.

POIBEAU, T., "Dealing with metonymic readings of named entities", *Annual Conference of the Cognitive Science Society*, vol. 4, pp. 1993–1998, 2006.

POIBEAU, T. & KOSSEIM, L., "Proper name extraction from non-journalistic texts", *Language and Computers*, vol. 37, pp. 144–157, 2001.

POPESCU, A.M. & ETZIONI, O., "Extracting product features and opinions from reviews", *Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 339–346, 2005.

PRINCETON UNIVERSITY, "WordNet", *About WordNet*, http://wordnet.princeton.edu/, accedido por última vez: February 22, 2013.

RAGHAVAN, H. & ALLAN, J., "Using soundex codes for indexing names in ASR documents", *Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval (Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies)*, pp. 22–27, 2004.

RAMAKRISHNAN, G., JOSHI, S., KHAITAN, S., & BALAKRISHNAN, S., "Optimization issues in inverted index-based entity annotation", *International Conference on Scalable Information Systems*, 2008.

RAU, L.F., "Extracting company names from text", *IEEE Conference on Artificial Intelligence Applications*, pp. 29–32, 1991.

REEVE, L. & HAN, H., "Survey of semantic annotation platforms", *ACM Symposium on Applied Computing*, pp. 1634–1638, 2005.

REEVE, L.H. & HAN, H., "CONANN: An online biomedical concept annotator", *Lecture Notes in Computer Science (Conference on Data Integration in the Life Sciences)*, vol. 4544, pp. 264–279, 2007.

RICHMAN, A. & SCHONE, P., "Mining wiki resources for multilingual named entity recognition", *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 1–9, 2008.

RILOFF, E. & JONES, R., "Learning dictionaries for information extraction by multi-level bootstrapping", *AAAI National Conference on Artificial Intelligence*, pp. 474–479, 1999.

RILOFF, E. & LEHNERT, W.G., "Automated dictionary construction for information extraction from text", *IEEE Conference on Artificial Intelligence for Applications*, pp. 93–99, 1993.

RINALDI, F., VASILAKOPOULOS, A., ZERVANOU, K., BERNARD, L., ZARRI, G.P., SLOT, H.B., TOUW, C. VAN DER, DANIEL-KING, M., UNDERWOOD, N., LISOWSKA, A., PLAS, L. VAN DER, DOWDALL, J., SAURON, V., SPILIOPOULOU, M., BRUNZEL, M., ELLMAN, J., ORPHANOS, G., MAVROUDAKIS, T., TARAVIRAS, S., HESS, M., KALJURAND, K., PERSIDIS, A., THEODOULIDIS, B., BLACK, B., MCNAUGHT, J., & KARANIKAS, H., "CAFETIERE: Conceptual Annotations for Facts, Events, Terms, Individual Entities, and RElations. Parmenides Technical Report TR-U4.3.1", *Department of Computation, University of Manchester Institute of Science and Technology*, 2005.

SALL, J., LEHMAN, A., STEPHENS, M., & CREIGHTON, L., "JMP start statistics: a guide to statistics and data analysis using JMP", (5th ed.), SAS Institute Inc., 2012.

SANG, E.F.T.K., "Introduction to the CoNLL-2002 shared task: language-independent named entity recognition", *Conference on Natural Language Learning*, pp. 155–158, 2002.

SANG, E.F.T.K. & MEULDER, F. DE, "Introduction to the CoNLL-2003 shared task: language-independent named entity recognition", *Conference on Natural Language Learning*, pp. 142–147, 2003.

SEKINE, S., GRISHMAN, R., & SHINNOU, H., "A decision tree method for finding and classifying names in Japanese texts", *Workshop on Very Large Corpora (Annual Meeting of the Association for Computational Linguistics and International Conference on Computational Linguistics)*, pp. 171–178, 1998.

SEKINE, S. & ISAHARA, H., "IREX: IR and IE Evaluation project in Japanese", *International Conference on Language Resources and Evaluation*, pp. 1475–1480, 2000.

SEKINE, S. & NOBATA, C., "Definition, dictionaries and tagger for extended named entity hierarchy", *Language Resources and Evaluation Conference*, pp. 1977–1980, 2004.

SETTLES, B., "Active Learning Literature Survey (Computer Sciences Technical Report 1648)", *University of Wisconsin-Madison*, 2010, disponible en: http://research.cs.wisc.edu/techreports/2009/TR1648.pdf.

SHEN, D., ZHANG, J., SU, J., ZHOU, G., & TAN, C.-L., "Multi-criteria-based active learning for named entity recognition", *Annual Meeting of the Association for Computational Linguistics*, 2004.

SHINYAMA, Y. & SEKINE, S., "Named entity discovery using comparable news articles", *International Conference on Computational Linguistics*, 2004.

SILVA, J.F. DA, KOZAREVA, Z., & LOPES, G.P., "Cluster analysis and classification of named entities", *Conference on Language Resources and Evaluation*, pp. 321–324, 2004.

SMITH, D.A., "Detecting and browsing events in unstructured text", *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 73–80, 2002.

SODERLAND, S., FISHER, D., ASELTINE, J., & LEHNERT, W., "CRYSTAL: inducing a conceptual dictionary", *International Joint Conferences on Artificial Intelligence*, 1995.

SODERLAND, S., "Learning information extraction rules for semi-structured and free text", *Machine Learning*, vol. 34, n.1, pp. 233–272, 1999.

SRIHARI, R. & LI, W., "Information extraction supported question answering", *Text REtrieval Conference*, pp. 185–196, 2000.

SUCHANEK, F.M., KASNECI, G., & WEIKUM, G., "Yago: a core of semantic knowledge", *International Conference on World Wide Web*, pp. 697–706, 2007.

SUCHANEK, F.M., M., S., & WEIKUM, G., "DS: Databases and Information Systems", *The YAGO-NAGA Project: Harvesting, Searching, and Ranking Knowledge from the Web. Max Planck Institut Informatik*, http://www.mpi-inf.mpg.de/yago-naga/, accedido por última vez: February 23, 2013.

THE UNICODE CONSORTIUM, "The Unicode Standard, Versión 6.1.0", The Unicode Consortium, 2012.

THOMPSON, C.A., CALIFF, M.E., & MOONEY, R.J., "Active learning for natural language parsing and information extraction", *International Conference on Machine Learning*, pp. 406–414, 1999.

TOMANEK, K., WERMTER, J., & HAHN, U., "An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data", *Join Conferences on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, vol. 3, pp. 486–495, 2007.

TROCHIM, W.M. & DONNELLY, J.P., "The research methods knowledge base", (3rd ed.), Atomic Dog Publishing, 2007.

TURMO, J. & AGENO, A., "Adaptive Information Extraction", *ACM Computing Surveys (CSUR)*, vol. 38, n.2, 2006.

URBANO, J., "Information retrieval meta-evaluation: challenges and opportunities in the music domain", *International Society for Music Information Retrieval Conference*, pp. 609–614, 2011.

UREN, V., CIMIANO, P., IRIA, J., HANDSCHUH, S., VARGAS-VERA, M., MOTTA, E., & CIRAVEGNA, F., "Semantic annotation for knowledge management: requirements and a survey of the state of the art", *Journal of Web Semantics*, vol. 4, n.1, pp. 14–28, 2006.

VILAIN, M., SU, J., & LUBAR, S., "Entity extraction is a boring solved problem: or is it?", *Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 181–184, 2007.

VLACHOS, A, "A stopping criterion for active learning", *Computer Speech & Language*, vol. 22, n.3, pp. 295–312, 2008.

VOORHEES, E.M., "The philosophy of information retrieval evaluation", *Lecture Notes in Computer Science (Evaluation of Cross-Language Information Retrieval Systems)*, vol. 2406, pp. 355–370, 2002.

WINKLER, W.E., "The state of record linkage and current research problems", 1999, disponible en: http://europepmc.org/abstract/CIT/255199.

WITTEN, I.H. & FRANK, E., "Data mining: practical machine learning tools and techniques", (2nd ed.), Morgan Kaufman, 2005.

WU, T. & POTTENGER, W.M., "A semi-supervised active learning algorithm for information extraction from textual data", *Journal of the American Society for Information Science and Technology*, vol. 56, n.3, pp. 258–271, 2005.

XUAN, W., DAI, M., MIREL, B., ATHEY, B., WATSON, S.J., & MENG, F., "Interactive Medline search engine utilizing biomedical concepts and data integration", *BioLINK SIG: Linking Literature, Information and Knowledge for Biology*, pp. 55–58, 2007.

YAROWSKY, D., "Unsupervised word sense disambiguation rivaling supervised methods", *Annual Meeting of the Association for Computational Linguistics*, pp. 189–196, 1995.

ZHU, J., UREN, V., & MOTTA, E., "ESpotter: adaptive named entity recognition for web browsing", *Lecture Notes in Computer Science (Professional Knowledge Management)*, vol. 3782, pp. 518–529, 2005.